Search and broadcast in stochastic environments, a biological perspective.

Lucas Boczkowski

Thèse de doctorat
de l'Université Sorbonne Paris Cité
Préparée à l'Université Paris Diderot

ÉCOLE DOCTORALE DE SCIENCES MATHÉMATIQUES DE PARIS CENTRE (ED 386)

Institut de Recherche en Informatique Fondamentale (IRIF)

Spécialité : Informatique

Search and Broadcast in Stochastic Environments, a Biological Perspective

---

Par :

Lucas BOCZKOWSKI

Directeurs de thèse : **Iordanis KERENIDIS** et **Amos KORMAN**

Soutenue publiquement à Paris le 30 novembre 2018 devant le jury constitué de :

Yehuda AFEK	PU	Tel-Aviv University	*Examinateur*
Pierre FRAIGNIAUD	DR	CNRS & IRIF	*Président du jury*
Iordanis KERENIDIS	DR	CNRS & IRIF	*Directeur*
Amos KORMAN	DR	CNRS & IRIF	*Directeur*
Claire MATHIEU	DR	CNRS & ENS	*Examinatrice*
Kurt MEHLHORN	PU	Max-Planck-Institut für Informatik	*Examinateur*
Luca TREVISAN	PU	U.C. Berkeley	*Rapporteur*
Roger WATTENHOFER	PU	ETH Zurich	*Rapporteur*

Titre : Recherche et diffusion d'informations dans un environnement bruité, une perspective biologique.

Résumé : Cette thèse s'articule autour de deux séries de travaux motivés par des expériences sur des fourmis. Bien qu'inspirés par la biologie, les modèles que nous développons utilisent une terminologie et une approche typique de l'informatique théorique.

Le premier modèle s'inspire du transport collaboratif de nourriture au sein de l'espèce *P. Longicornis*. Certains aspects fondamentaux du processus peuvent être décrits par un problème de recherche sur un graphe augmenté d'un certain type d'indications bruitées à chaque noeud. Ces indications représentent de courtes traces de phéromones déposées devant l'objet transporté afin de faciliter la navigation. Dans cette thèse, nous donnons une analyse complète du problème lorsque le graphe sous-jacent est un arbre, une hypothèse pertinente dans un cadre informatique. En particulier, notre modèle peut être vu comme une généralisation de la recherche binaire aux arbres en présence de bruit. De manière surprenante, les comportements des algorithmes optimaux dans ce cadre diffèrent suivant le type de garantie que l'on étudie : convergence en moyenne ou avec grande probabilité.

Le deuxième modèle présenté dans cette thèse décrit la diffusion d'informations au sein de fourmis du désert. Dans notre modèle, les échanges ont lieu uniformément au hasard, et sont sujets à du bruit. Nous prouvons une borne inférieure sur le nombre d'interactions requis en fonction de la taille du groupe. La borne, de même que les hypothèses du modèle, semblent compatible avec les données expérimentales. Une conséquence théorique de ce résultat est une séparation dans ce cadre des variantes PUSH et PULL pour le problème du broadcast avec bruit. Nous étudions aussi une version du problème avec des garanties de convergence plus fortes. Dans ce cas, le problème peut être résolu efficacement, même si les messages échangés au cours de chaque interaction sont très limités.

Mots clefs : Algorithmes biologiques - Dissémination d'informations - Recherche robuste - Complexité moyenne - Erreurs permanentes - Communication bruitée

Title: Search and broadcast in stochastic environments, a biological perspective.

Abstract: This thesis is built around two series of works, each motivated by experiments on ants. We derive and analyse new models, that use computer science concepts and methodology, despite their biological roots and motivation.

The first model studied in this thesis takes its inspiration in collaborative transport of food in the *P. Longicornis* species. We find that some key aspects of the process are well described by a graph search problem with noisy advice. The advice corresponds to characteristic short scent marks laid in front of the load in order to facilitate its navigation. In this thesis, we provide detailed analysis of the model on trees, which are relevant graph structures from a computer science standpoint. In particular our model may be viewed as a noisy extension of binary search to trees. Tight results in expectation and high probability are derived with matching upper and lower bounds. Interestingly, there is a sharp phase transition phenomenon for the expected runtime, but not when the algorithms are only required to succeed with high probability.

The second model we work with was initially designed to capture information broadcast amongst desert ants. The model uses a stochastic meeting pattern and noise in the interactions, in a way that matches experimental data. Within this theoretical model, we present in this document a strong lower bound on the number of interactions required before information can be spread reliably. Experimentally, we see that the time required for the recruitment process of even few ants increases sharply with the group size, in accordance with our result. A theoretical consequence of the lower bound is a separation between the uniform noisy PUSH and PULL models of interaction. We also study a close variant of broadcast, without noise this time but under more strict convergence requirements and show that in this case, the problem can be solved efficiently, even with very limited exchange of information on each interaction.

Keywords: Biological algorithms - Information dissemination - Fault tolerant search - Average case analysis - Permanent faults - Noisy communication

Acknowledgements

This thesis would have little to do with its current form had I not benefited from so many lucky encounters and fruitful interactions over the past three years, starting with my two outstanding advisers, whose guidance and support will be hard to match if I do another PhD.

Amos, your advice although not delivered in the form of pheromones was precious to reach the end of this thesis. The dedication you put in your work, and the confidence with which you manoeuvred through the obstacles of research made a lasting impression on me. Thank you for involving me in your unique scientific enterprise, connecting the algorithms and the ants, between Paris and Tel Aviv. Needless to say you changed my perception of many aspects of life beyond computer science (in a good way).

Iordanis, we worked together mostly at the outset of this journey, but you were always a great support along the way. I was always impressed by the breadth and acuteness of your views on the field. The class you gave when I was a Master's student was a pivotal moment for me, which motivated me to engage in a PhD.

I am very grateful to Luca Trevisan and Roger Wattenhofer for taking the time to review this manuscript and giving me very insightful feedback. I would also like to thank the members of the jury Yehuda Afek, Pierre Fraigniaud, Claire Mathieu and Kurt Mehlhorn for their interest in my work, and for being present at my defense.

In my experience, research was everything but a solitary exercise. I would like to warmly thank Emanuele Natale and Yoav Rodeh for the time spent together around the globe (Università Tor Vergata, the Max Planck Institute and the Weizmann Institute), increasing my appreciation of rigor and calm respectively and of course sharing so many ideas. This thesis would lack a great deal of motivation if it were not for Ofer Feinerman's ant lab at the Weizmann Institute. Thank you for making every visit such an interesting experience. The collaboration with Uriel Feige has been a great drive to develop more results on the advice model. At IRIF, I have also learned a lot from collaborating with Adrian Kosowski and Frédéric Magniez. More recently, I enjoyed working with Brieuc Guinard and Ami Paz.

Ma thèse est largement tributaire de l'excellente atmosphère qui règne à l'Université Paris 7 et plus particulièrement à l'IRIF, laboratoire qui m'a accueilli au cours de ces trois ans. La vie quotidienne y fut très agréable notamment grâce à Guillaume et Alexandre dont j'eus l'honneur de partager le bureau, mais aussi Alex, Laurent, Mengchuan, Yassine, Mathieu, Alkida, Dennis, Florent, Pierre et tous les autres doctorants, collègues devenus amis.

Ces remerciements s'étendent bien sûr aux membres de l'administration, dont le travail a grandement facilité ma vie de chercheur. Un grand merci donc à Noëlle, Odile, Etienne, Dieneba, Amina, Houy et Laïfa.

Je suis également reconnaissant à l'école républicaine et ses différents professeurs qui m'ont donné le goût des mathématiques, ainsi qu'à Marc Lelarge pour m'avoir encadré en master de probabilités et donné le goût des sujets inter-disciplinaires. In the past, Yuval Peres and Rüdiger Urbanke have been great internship mentors.

J'aimerais aussi remercier mes amis : scientifiques, littéraires, littéraires post-coloniaux, coureurs de fond, pousseurs de bois, pousseurs de luges, chefs cuistots, végétariens en période probatoire, végans endurcis sauf en Russie, défenseurs professionnels de la Nature, cinéastes, cinéphiles et assimilés, ceux qui m'ont fait grabataire ou cadavre pour une cause artistique, ceux qui m'ont fait passer de si bons moments autour d'un dîner à Paris, dans leur chalet en Savoie (vous êtes plusieurs) ou dans toute autre propriété familiale, ceux qui étaient là pour le dernier déménagement (vous n'êtes pas plusieurs), ceux que j'ai été voir à l'étranger et ceux qui m'attendent encore : Anatole, Anne, Antoine, Arthur B., Arthur L.B., Camille, Charles F., Charles M., Christian, Christophe, Clément M., Clément P., Emmanuel, Félix, Gabrielle, Gary, Giulia, Ines, Kenza, Lisa, Louis, Lucas, Marianne Ha., Marianne Hi., Maxime, Mélina, Minh-Tu, Nicolas, Pauline, Pierre B., Pierre D., Raphaël, Rebecca, Simon, Sophie, Thomas, Victor.

The Argentinean, the Swedish, and the Israeli branch of my family, all have my gratitude for their continuous hospitality, particularly during the making of this work. I am honored to see some of their members coming to my defense. J'ai également une pensée spéciale pour la branche Blésoise[1], dont je me réjouis de la présence à ma soutenance et au-delà. Je salue enfin les oncles et tantes de coeur, amis de mes parents qui me font aussi la joie d'être présents.

Je ne dirai que peu de choses ici à propos de ceux à qui je dois le plus. Merci à Jorge et Diana, mes parents pour leur écoute et leur soutien inestimable. L'intransigeance intellectuelle vivifiante de mon frère Octave a aussi beaucoup compté. J'ai apprécié son soutien à distance, ainsi que celui plus récent de Nicole.

Merci enfin et surtout à Elsa, sans qui je n'envisagerais rien de tout le reste.

[1] en incluant Betsy que je remercie en particulier pour la relecture de ces remerciements

Contents

Chapter 1

Introduction

1.1 Summary

This thesis is articulated around two series of works at the junction between computer science and biology. The first series revolves around a new kind of graph search problem, while the second is about broadcast.

The trajectory we follow is similar in the two cases, and summarized in Figure 1.1. Our original scientific approach bridges theoretical algorithm analysis and experimental biology, with a high level goal of reinforcing the interactions between the two fields. Experiments on ants performed at Ofer Feinerman's lab at the Weizmann Institute provide inspiration and a reference point that drive the design of new theoretical models. The models are accurate enough to capture some aspects of the biological algorithmic process at hand, but at the same time simple enough to lend themselves to rigorous mathematical analysis and to encompass a broad array of biological phenomena. By understanding which tasks are efficiently solvable in our framework, we hope to obtain new insights on the experiments and the associated biological phenomena. We also find relevance for our works in more standard computer science contexts, so we refine the analysis in that direction, increasing the technical depth at the cost of losing the connection with biology.

Both our broadcast and search models are motivated by experiments on food foraging, albeit with different species of ants. The first setup uses ants of the species *Paratrechina longicornis* commonly referred to as "Crazy ants". These tiny ants occasionally carry large insects back to their nest, by working in groups. The navigation is a complicated mechanism, due to the large number of individuals that are involved, and the route not being well established. Ants that are not carrying the load are able to help routing it by laying short scent marks. However this guiding mechanism is not without faults. The individual ants signal a path that is good from their perspective, but that may be too narrow for the larger item being carried.

In [71], we studied the kind of trail that is used by crazy ants. It presents characteristics that were unreported before for any kind of ant. To capture the essence of the process, we

introduce a simplified model that corresponds to a search problem on a graph with unreliable advice. The advice, modeling pheromones, comes in the form of pointers located at each node. The pointers indicate a step on a shortest path to the target node, but with some probability they are misleading and then point to a randomly chosen neighbor. Crucially, the pointers are fixed throughout the execution, and cannot be re-sampled. This yields a new model of noise, leading to new algorithmic challenges and tailored techniques to overcome them. We analyzed the model on the line and two dimensional grids first [71] and showed connections with the experiments on ants. In subsequent work [28, 31] geared towards the computer science community, we carried an analysis on trees, having in mind tree-like data structures. In this manuscript, we present the works [28] and [31] in Chapters 2 and 3.

In the second line of research, the experiments were carried on *Cataglyphis niger*, a desert ant species. When a forager ant finds a large amount of food outside the nest, it tries to share that information with her peers who remained inside the nest, in order to recruit more ants to deliver the food. Desert ants do not rely on pheromones to communicate. Instead they use physical contact to convey information.

Our approach consists in viewing recruitment as a broadcast problem with a stochastic meeting pattern. The model incorporates noise in the interactions, in a way that matches the data gathered from experiments. Within this theoretical model for broadcast with noisy interactions, we are able to show a strong lower bound on the number of interactions required before information can be spread reliably to at least another individual. Although this is a negative result, it sheds a new light on the experiments, by giving a new counterintuitive prediction. Indeed, we see that the time required for the recruitment process of even few ants increases sharply with the group size, in accordance with our result. Due to its generality, the lower bound may also be relevant to other biological systems. In Chapter 4, based on [24], we present the model and experiments in more details, and the associated lower bound. In Chapter 5, based on [29], we study a close variant of broadcast, without noise this time but under more strict convergence requirements.

1.2 Organization of the Introduction

A general background is given in Section 1.3. The works presented in this manuscript and their connection to biology are explained in greater detail in Sections 1.4 and 1.5. In Section 1.6, we expand on Section 1.3 by presenting other attempts to bridge computer science and biology and more related work. In Section 1.7 two of my works that are outside of the scope of this manuscript are briefly presented.

1.3 Background

Throughout its short history, computer science has already approached biology on several occasions, and has had a profound impact on it. Perhaps the best established area of interaction

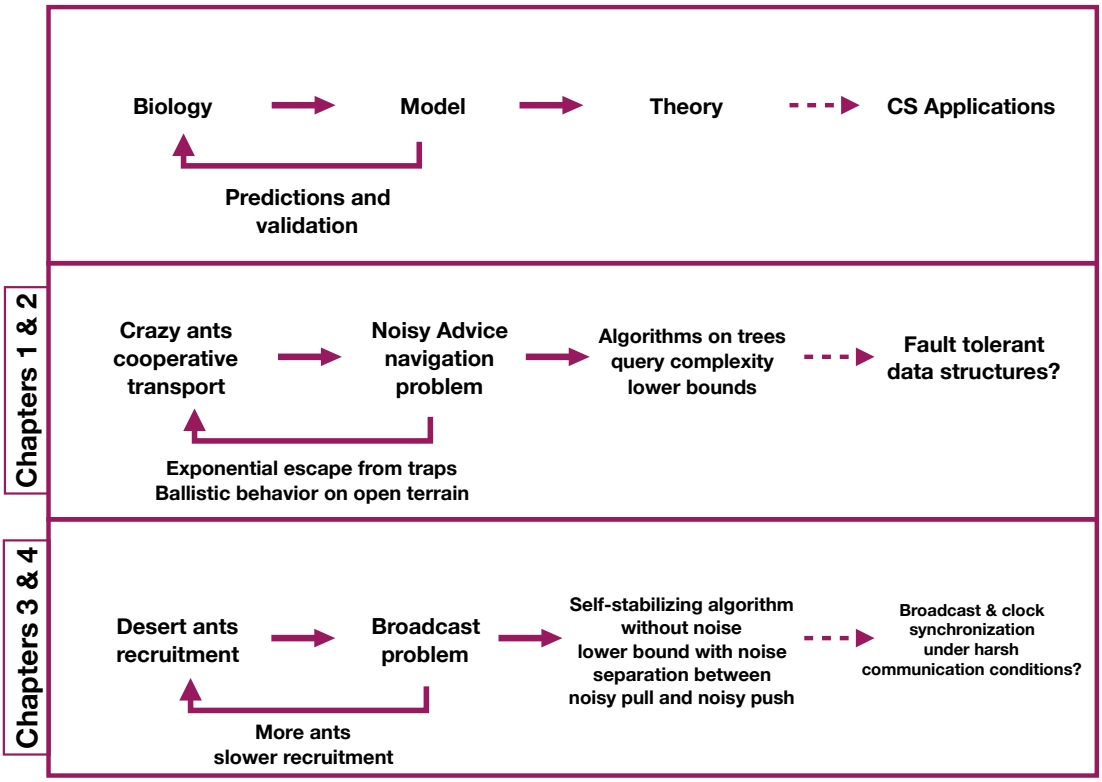

Figure 1.1: The framework behind the works presented in this manuscript.

between the two disciplines is bio-informatics. The field of bio-informatics leverages computational tools to manipulate and analyze biological data, for instance nucleic acid sequences. On the theoretical side, in 1956 already, Von Neumann wrote an essay entitled *The Computer and the Brain*, which examines similarities and differences between the functioning of the brain and that of computers.

Analyzing algorithms in the natural world requires handling many unknowns. The setting is often not well defined, and sometimes the algorithm itself is not well specified either. In line with the spirit of comparing biological processes to computational ones, several works have drawn inspiration from biology to derive new algorithmic ideas. This fruitful approach was pioneered in distributed computing through the Beeping model [1, 2], its companion "Stone age" computing model [58] and population protocols [7]. More information on these interesting lines of research is given in Section 1.6. Schematically, restrictions on communication or computing power are added to standard distributed computing models to form new models, that are better suited to describe biological systems. With such a bio-inspired model in hand, it is possible

to study the feasibility of a given computational task, such as broadcasting or computing a Maximal Independent Set. Another approach consists in studying an algorithm that is thought to be relevant to describe some biological process. A prominent example in this category is the work by Chazelle on flocking [39]. In all these research lines however, it is rare to see close collaborations between experimental biologists and computer scientists, [2] being a notable exception. Thus, the theoretical findings are not necessarily confronted to the experiments that inspired them.

Mathematical modeling has already been used in other disciplines to capture the essence of biological phenomena. For the particular case of social insects and ants in particular, there is a rich literature in the field of theoretical ecology. The models proposed by ecologists since the 60's draw on economic reasoning to characterize the optimality of decisions made by ants (where to locate the nest, how to forage etc.). More recently, physicists have also proposed models inspired by biology, importing techniques and tools from their field. Their models are often based on continuous objects such as PDE's, whereas their discrete counterparts are favored in computer science. A more fundamental difference between the two fields is that in computer science, the algorithm is systematically decoupled from the computational model, and viewed as one way amongst others to achieve a given task. [1]

1.4 Searching with Noisy Local Advice

In this section, the Noisy Advice Model of graph search is introduced. The model has a biological motivation: it is closely related to experiments on ants, presented and analyzed in [71]. The results presented in Section 1.4.4 are taken from the works [28] and [31], that are the basis of Chapters 2 and 3 in this manuscript.

1.4.1 Biological Setup

P. longicornis ants are a very common ant species also known as "Crazy ants". They are small: adult specimens measure approximately 2mm. When they find a large food item while foraging, they can engage in a cooperative transport process where several individuals join forces to bring the item to the nest (see Figure 1.2). This is a very intricate mechanism, that is interesting as an example of a complex collective behavior. Moving the load requires the ants to master a scale that is beyond their own. Due to the difference in size between the ants and the load they carry, there is a gap between the ants perspective and the knowledge required for the load navigation. This can potentially lead to conflicts, or inefficiencies.

An important step towards understanding the navigation process was made when Ofer Feinerman and his group at the Weizmann Institute were able to show empirical evidence for a

[1] Famous examples of computational models include the automata, Turing machine, the streaming model, communication complexity models etc. In each of those, several algorithms or protocols exist to solve a particular task, with different performance guarantees. The performance is measured with respect to a relevant computational resource such as time, space, communication.

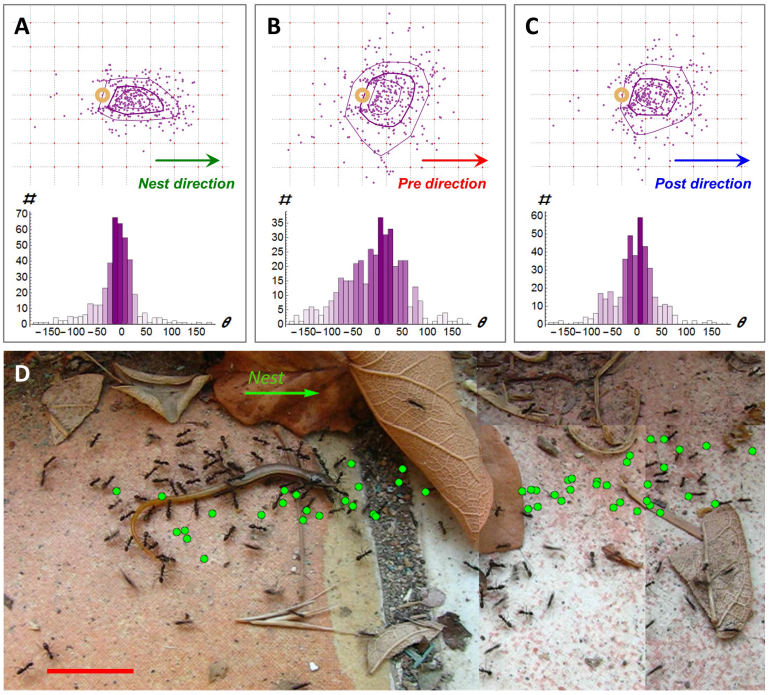

Figure 1.2: (A-C). Distribution of marking events (N = 408) during a specific example of 113 cm of cooperative transport. Upper panels show the spatial distribution of scent marks (purple dots) locations, upon marking, in a moving frame of reference that is attached to the center of the transported load. The x-axis of this reference frame points towards the nest (a) or in the direction of the load movement in the 2 s that immediately proceed (b) or follow (c) the time this mark was deposited. Purple lines indicate quartile polytopes. Bottom panels: Angular distribution of the same data points. (D) Cooperative transport of a large prey item in a natural environment. Green dots denote scent marks. Red bars denote 2 cm. This figure appeared as Figure 3 in [71].

new kind of scent marks used during the retrieval stage of cooperative transport. These marks are laid by individual ants that are not carrying the load, and can be understood as guiding instructions for the ants carrying the load. This common function of the pheromone is however associated here to some unique features that also distinguish them from previously described ant trails[2].

1. They are short (median bout length is less than 10 cm) and volatile.

2. They typically convey precise information about the nest direction but they can be misleading. This is interpreted as a consequence of the impossibility for single ants to detect

[2]All statements about ant behavior given here are based on numerical evidence provided by Feinerman's team. See [71] for details

which paths are accessible for the much larger load being carried.

3. Empirically, it can be shown that the load typically follows the scents close to it, but also occasionally deviates from them.

4. A trail composed of short scents is dynamically created in front of the load as it moves. That is, everywhere the load goes, even if it deviates from a shortest path to the nest, it keeps receiving navigation instructions.

In [71], we propose a model for the navigation process that incorporates the main macroscopic aspects of the pheromone guiding mechanism. For this, we use the language of discrete graphs, and view the short scent marks as *local advice*.

1.4.2 The Noisy Advice model

We now introduce the Noisy Advice Model. Anticipating the following sections, the model is given in a somewhat general form, including several variants of interest. The link between the abstract model and the experiment is clarified in Section 1.4.3.

Let G be an undirected bi-directional graph, where a treasure has been hidden at some node $\tau \in G$. A searcher aiming to find τ interacts with G in the following way. The searcher can probe any node v and learn if $\tau = v$. If not, it is given an indication, called *advice* in the form of a node u, amongst the neighbors of v. If the advice is valid, it should satisfy $d(u, \tau) = d(v, \tau) - 1$, where $d(\cdot, \cdot)$ is the hop distance on the graph. In words, u is a possible first step on a shortest path from v to τ.

Noise assumptions. Initially, each node is declared faulty with some fixed probability $q \in (0, 1)$. If a node v is faulty, the advice it holds is either a random neighbor or an adversarially chosen one. Importantly, the advice at each node is fixed throughout the execution of the algorithm.

The most important aspect of our noise model is that advice is drawn once and for all. When querying several times the same node, the answer remains the same.

Semi-adversarial variant. The purely-probabilistic Noisy Advice Model is well suited to describe the ants' navigation, as we explain below. Instead, in the semi-adversarial variant, first, the adversary specifies the faulty advice w for each node u in case u is faulty, and then the environment samples which node is faulty and which is sound. Therefore, the adversary does not get to choose if a node is faulty, but rather what advice it would show if it is chosen (randomly) as faulty.

Complexity measures. We mainly keep track of two resources. Our algorithms are either efficient with respect to the number of moves made (edges traversed) or the number of queries made before finding the treasure. Note that the number of advice queries is always smaller than the number of moves.

Convergence requirements. We produced three related works on this model. In the first two [71, 31], we were concerned about finding efficient algorithms in expectation over the faulty locations, whereas in the third one [28], we studied the problem under high probability requirements. In this manuscript, works are grouped according to the complexity measure (moves VS queries) rather than the kind of guarantee on the running time.

1.4.3 Connection to Experiments and First Results

The agent moving along the edges of the graph models the moving load seeking the nest/target τ. Local advice models the guiding scent marks laid by individual non-carrying ants[3]. The crucial assumption that advice is permanent is motivated by the fact that the environment does not change significantly during the motion of the load. Hence, if the scents are misleading at a given point in time and space, it is likely to remain misleading at the same point in space, later in time. Considering probabilistic rather than worst case faults models the fact that the terrains ants face are not designed by an adversary.

Probabilistic Following. Empirical analysis carried by Feinerman's team revealed that the algorithm performed by the ants can be approximated by a randomized strategy, which we term *Probabilistic Following*. This is a memory-less algorithms that consists in following the advice proposed at each step with some probability $\lambda \in (0, 1)$ and making a random move otherwise.

In the paper [71] (results not included in this manuscript), we analyzed the Probabilistic Following model over two kinds of graphs. Recall that q is the mistake parameter, it corresponds to the probability of each advice being faulty. On 2-dimensional grids, describing an open terrain, we were able to show that if the mistake parameter q is sufficiently low, under an appropriate choice of λ, the probabilistic following strategy is ballistic. That is, it moves in the direction of the nest at a constant speed. This analysis heavily relies on results from the theory of random walks in random environment (RWRE), of which the process we study is a particular example.

On the line, we show that the Probabilistic Following strategy has a linear speed if $q < \frac{1}{2}$ and an exponential speed if $q > \frac{1}{2}$. The proofs are also adaptations of known RWRE results.

Empirical confirmation. In the Navigation was studied on an open terrain (corresponding to a grid in our model). As the theory predicts, even if the scents do not always lead to the nest, the travel time scales linearly with the distance to the nest. In contrast, it is possible to design obstacles that will require the ants exponential time to bypass (exponential in the size of the obstacle, see Figure 1.3C). The idea is to design a big line-shaped wall with a narrow

[3]The scent marks are volatile, so once the load follows some advice scent it is likely to have disappeared if it ever comes to the same spot again. At that moment new scent (approximately identical to the first one) will be dropped and so forth. This means we can safely assume advice is directed, even if in real life there is no evidence for directionality in the scent marks themselves.

slit at the center. The slit allows for single ants to pass. If no ant comes from the side, all the scents point towards the slit and are thus misleading. The load will stay stuck trying to pass through the slit, when the correct decision would be to go around the wall, either through the left or right. This situation can be seen as a line with $q = 1$ where indeed the model predicts exponential time to reach the endpoint. It is worth noting the measurement on an open terrain and the design of the obstacles were guided by our theoretical findings.

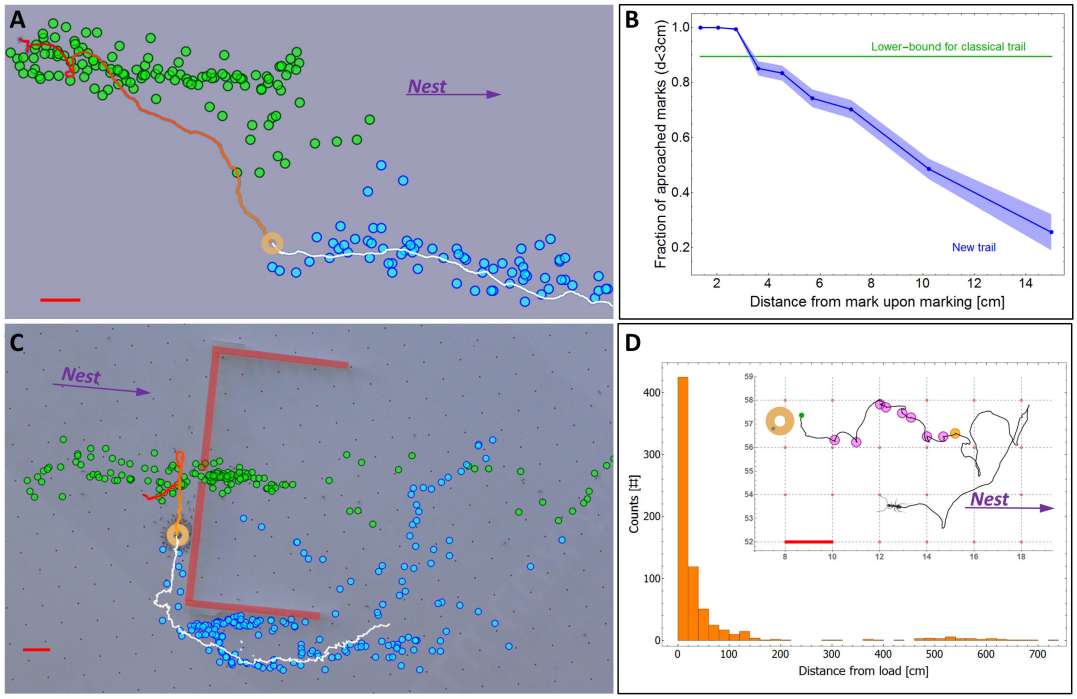

Figure 1.3: (A) In an obstacle-free environment, the load loses the scent trail which then reforms in front of it. Green dots indicate the position of scent marks produced before the load reached the position indicated on the image (ring). Blue dots indicate scent marks laid after this time. Solid line marks the load's trajectory. (B) The probability that the load eventually approaches (to less than 3 cm) a scent mark as a function of the distance between them at the moment of marking. For comparison, the green line depicts the corresponding curve for a classical ant trail (C) Cooperative transport while bypassing an obstacle (thick red lines) with a slit. Load position and marking colors as in panel (a). (D) Distribution of single ant marking bout lengths defined as the distance between the load and the furthest mark in a marking sequence. The inset shows a typical marking bout of nine marks (discs). Furthest mark is denoted in orange. Red bars denote 2 cm. This figure appeared as Figure 3 in [71].

	Upper Bound			Lower Bound		
	N.R.	Moves	Queries	N.R.	Moves	Queries
$\mathbb{E}$	$q \ll \frac{1}{\sqrt{\Delta}}$	$O(d\sqrt{\Delta})$	$\tilde{O}(\sqrt{\Delta}\log n)$	$q \gg \frac{1}{\sqrt{\Delta}}$	$e^{\Omega(d)}$	$n^{\Omega(1)}$
h.p.	$q = \Delta^{-\varepsilon}$	$d^{O(\varepsilon^{-1})}$	$(\log n)^{O(\varepsilon^{-1})}$	$q = \Delta^{-\varepsilon}$	$d^{\Omega(\varepsilon^{-1})}$	$(\log n)^{\Omega(\varepsilon^{-1})}$

Figure 1.4: **Purely-random variant.** A summary of the results presented in Chapters 2 and 3, in simplified form. The symbol $\mathbb{E}$ indicates results that hold in expectation, while the shorthand *h.p.* stands for *high probability*. The precise conditions behind the symbol $\ll$ will be clarified later. We write N.R. in place of Noise Regime.

	Upper Bound			Lower Bound		
	N.R.	Moves	Queries	N.R.	Moves	Queries
$\mathbb{E}$	$q \ll \frac{1}{\Delta}$	$O(d\sqrt{\Delta})$	–	$q \gg \frac{1}{\Delta}$	$e^{\Omega(d)}$	$n^{\Omega(1)}$
h.p.	$q = \Delta^{-\varepsilon}$	$d^{O(\varepsilon^{-1})}$	$(\log n)^{O(\varepsilon^{-1})}$	$q = \Delta^{-\varepsilon}$	$d^{\Omega(\varepsilon^{-1})}$	$(\log n)^{\Omega(\varepsilon^{-1})}$

Figure 1.5: **Semi-adversarial variant.** A summary of the results presented in Chapters 2 and 3, in simplified form, for the semi-adversarial variant.

1.4.4 Theoretical Results

Later, the analysis of the advice model was deepened by studying more complicated algorithms than probabilistic following [28, 31]. We also moved the focus to trees, which is a prominent graph family in computer science, due to important data structure applications. These results are presented in Chapters 2 and 3.

Our results on trees are summarized in Table 1.4 for the standard purely-random variant and Table 1.5 for the semi-adversarial variant. As mentioned previously we study both the number of edge traversals needed before finding the target and the number of advice queries, in expectation and with high probability. Chapter 2 deals with walking algorithms, while Chapter 3 is about their query counterparts. Interestingly, when considering high-probability algorithms, the semi-adversarial variant does not introduce a difference in the threshold for efficient algorithms. Another interesting point is that Probabilistic Following (the ants algorithm) turns out to be optimal for walking algorithms in the semi-adversarial variant.

1.4.5 Different Guarantees - Different Approaches

Each type of complexity measure and convergence guarantee is linked to different algorithmic ideas. Let us briefly present the strategies underlying our algorithms.

Moving fast in expectation. The crucial notion in the design of the algorithm is a carefully crafted *scoring* function. It assigns to each node a number between 0 and 1, which we interpret as a likelihood to lead to the treasure, given the advice seen so far by the searcher. At a given point in time, the algorithm has already seen a connected component of nodes of the tree. It considers the frontier of this connected component and decides to visit the node with the highest score on this frontier. Visiting this node yields one extra piece of advice, which in turn allows to update of the scores. The analysis then involves estimating the expected number of nodes with a better score than the actual node leading to τ, at each level of the tree.

The approach we follow is akin to Bayesian search heuristics, where a prior on the location of the object being searched is updated after every move using Bayes rule. Indeed our scoring function is essentially a likelihood, computed with respect to a certain prior. This is perhaps surprising for we do not assume any distribution on the placement of the treasure, but rather consider it is placed by an adversary. The big challenge is finding the right prior distribution that in some sense represents the adversary.

Moving fast with high probability. The approach taken here is slightly different. The key concept, rather than being a scoring function, is a notion of node *fitness*, which can also be viewed as a scoring function taking only binary values. Essentially, a node is declared fit if it has many pointers to it on the path coming from the root. This is good evidence that the node is either on the path to the treasure, or at least not too far from it. The idea is to explore the component of fit nodes to which the root, i.e., the starting point, belongs. If the component contains the treasure and few nodes outside the root to treasure path, then the treasure is found fast. With the appropriate formalization of fitness, we can show that these sufficient conditions for efficient search hold with high probability.

Queries. The number of queries is measured with respect to the total size of the tree n, rather than the distance to the treasure d. We build on a separator based-scheme, that would find the treasure using $O(\log n)$ queries in the absence of faults. Since the advice is not fully reliable, a mechanism is needed to catch the error in case the advice at a separator is faulty. Probing the whole neighborhood around a separator would be too costly. Hence, we resort to a local exploration which actually corresponds to a walking algorithm as the ones described in the previous paragraph. The local exploration corrects potential errors at all $O(\log n)$ separators on the way to the treasure, with high probability, thus leading to efficient algorithms in the high probability setting.

Local exploration may however fail due to a large quantity of errors around a separator. A simple remedy is to settle this case by querying the whole tree. This does not completely break

the expected runtime, as it yields a $O(\log^2 n)$ algorithm. To derive our best query-algorithm, we use two scales of local exploration. The biggest scale is used as a fallback in case local exploration at the first scale fails.

Randomized VS Semi-adversarial. In some cases, the same algorithms that work in the purely random variant, that is, when the faulty nodes have a uniformly chosen advice, also perform well in the semi-adversarial variant. This holds for our walking algorithms in the high-probability case.

When considering algorithms with a good performance in expectation, the adversary choosing the direction of advice at faulty nodes is able to delay the phase transition from $q = \frac{1}{\sqrt{\Delta}}$ to $q = \frac{1}{\Delta}$. This means that when the noise parameter q is greater than $\frac{1}{\Delta}$, any algorithm has exponential expected walking complexity in the semi-adversarial variant.

It turns out that Probabilistic Following, the memoryless algorithm we gave as a good approximation to the ants behavior, is also optimal in that setting, for it reaches the target in linear time when $q < \frac{1}{\Delta}$, even in the semi-adversarial variant.

1.5 Broadcast under Harsh Communication Conditions

1.5.1 Biological Setup

Cataglyphis niger ants are a kind of scavenger ants living in deserts or arid environments. They typically look for food outside their nest individually. If an oversized food item is found in the vicinity of the nest, a forager may try to recruit other individuals to help her carry it back to the nest. For desert ants however, cooperative transport is more rare than for crazy ants[4]. It involves fewer individuals and no pheromone. It is thought that desert ants do not use pheromones because the hot environments they live in would make it evaporate too fast.

We are interested in that moment where a single ant holds a piece of information (the presence of food in the vicinity of the nest, and perhaps its location) and wishes to share it with her peers. *C. niger* ants primarily communicate by bumping into each other, and not, as said previously, using pheromones.

Experiments were carried at the Weizmann Institute of Science by Feinerman's team. Groups of moderate size (up to 10 individuals) were exposed to a blocked food item. An ant, upon discovering it and failing to move it on her own returns to seek help amongst her peers. In that setup recruitment happens in the small area of the nest's entrance chamber. Within this confined area, the interactions between ants are nearly uniform [102], such that an ant cannot control which of her nest mates she meets next. Additionally, it has been shown that recruitment in *Cataglyphis niger* ants relies on rudimentary alerting interactions [52, 85] which are subject to high levels of noise [106]. Furthermore, the responses to a recruiting ant and to an ant that is randomly moving in the nest are extremely similar [106]. We consider that

[4]It was documented only recently [6]

the forager succeeded in alerting about the presence of a food source when, after her return to the nest, at least one other ant went outside as well.

1.5.2 Bit Dissemination in the Pull Model

To approach this experimental setup, we use the language of broadcast, also known as bit dissemination or information spreading. Below we present the main assumptions we work with. The model we work with retains a certain degree of generality. Although it is well suited to describe recruitment in desert ants, we hope that it is simple enough to describe other biological setups as well. The general underlying principle is to gain understanding into the conditions that make bit dissemination feasible.

The problem. The word broadcast is used interchangeably with bit dissemination, and rumor spreading. The setup is as follows. Within a population of n agents, a source holds a value $b \in \{0, 1\}$ it wishes to share with the rest of the group. Sometimes, we also extend this assuming there are several sources, each with their fixed input bit. In that case, we want a non-source agent to learn the majority of all source inputs.

Agents. Since we are looking for a lower bound, we are rather liberal and assume agents are able to make arbitrary computations. This is convenient because we do not want to make strong assumptions on the ants inner functioning. The limitations come from the communication setting. Each agent holds a message, that can be updated at any time, but it is subject to noise.

Noisy pull interactions. We assume the model of interaction is the uniform $\mathcal{PULL}$ model. This means that at each time slot, each agent synchronously samples one other agent and reads the message they are displaying. Interactions are subject to noise. We assume each message sent can be transformed into any other message with some small probability $\delta > 0$.

Convergence guarantees. Since the interactions are randomized, we cannot hope to guarantee agents will recover the value kept by the source on every run of the protocol, within a given time bound. Instead, we ask for convergence with non trivial probability, usually meaning with probability $\frac{2}{3}$.

Number of interactions. The main resource we keep track of is time: how many pairwise interactions or parallel rounds are needed before convergence. If there is no noise, and if we do not enforce self-stabilization, a simple protocol works in $O(\log n)$ parallel rounds. At a high level, this is because on each round, the fractioned of informed agents increases by a constant multiplicative factor (in expectation, at least). Hence, usually, when working with models like ours, the goal is to achieve logarithmic convergence time.

1.5.3 Theoretical Results and Connection with Experiments

Within our theoretical framework, we are able to show the following theorem, stated here informally.

Theorem 1.1

No protocol can solve the broadcast problem in the uniform $\mathcal{PULL}$ model in less than $\Omega(n)$ rounds. The constant in the term $\Omega(n)$ hides a dependency on the noise parameter δ.

There are at least three ways to approach this negative result. To begin with, it is in line with the experimental observations made on *C. niger ants*. The interactions between ants are well described by our random meeting pattern (Figure 1.6(b)). Ants encode information in the speed they use when bumping into each other. The response to different messages (here bumps of different speeds) are statistically hard to distinguish (Figure 1.6(c)). We take this as evidence that the interactions are noisy. In fact, the lower bound itself can be taken as another validation that our model is a reasonable abstraction for the biological system at hand. The recruitment time deteriorates with group size, which is in accordance with our lower bound. See Figure 1.6(d) for details.

A more indirect way to gain biological insights from Theorem 1.1 is by taking its contrapositive. This leads to the following general meta-statement: in order to achieve efficient rumor spreading, a system must exhibit either some degree of structural stability or, alternatively, some facet of the communication which is immune to noise. This interesting insight can be confronted with different examples. For instance the interactions between synapses is known to be noisy, yet the brain as a whole has a reliable behavior. It appears a key factor that allows for such global efficiency despite unreliability at the synaptic level is the structural stability of the cell network.

All assumptions in the lower bound are necessary for the result to hold. If we replace the $\mathcal{PULL}$ model by the $\mathcal{PUSH}$ model, the very same problem we studied in Section 1.5.2 can be solved efficiently [63], even if the messages are corrupted by noise. The $\mathcal{PUSH}$ model is different in that, agents seeking to share a piece of information deliberately engage in an interaction, and may otherwise stay silent. It should be noted that the very act of engaging into an interaction is not altered by noise in the $\mathcal{PUSH}$ model. The lower bound of Theorem 1.1 may thus also be interpreted as an exponential separation between the $\mathcal{PULL}$ and $\mathcal{PUSH}$ models with noise.

1.5.4 A Self Stabilizing Solution without Noise

Removing the noise assumption makes our problem efficiently solvable, even in the $\mathcal{PULL}$ model. We went further in that direction and added other constraints to see what could still be done with purely pull interactions. The changes with respect to the previous assumptions

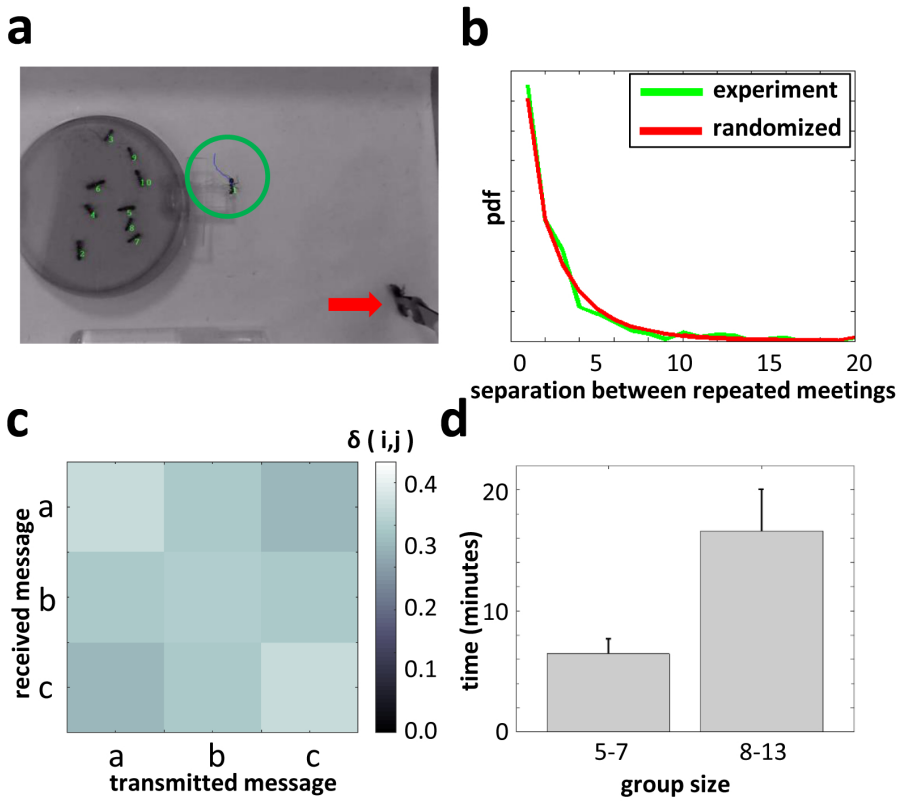

Figure 1.6: **Unreliable communication and slow recruitment by desert ant (*Cataglyphis niger*).** **a.** The experimental setup. The recruiter ant (circled) returns to the nest's entrance chamber (dark, 9cm diameter, disc) after finding the immobilized food item (arrow). Group size is ten. **b.** A *probability density function (pdf)* of the number of interactions that an ant experiences before meeting the same ant twice. The *pdf* is compared to uniform randomized interaction pattern. Data summarizes $N = 671$ interactions from seven experiments with a group size of 6 ants. **c.** Interactions of stationary ants with moving ants were classified into three different messages ('a', 'b', 'c') depending on the ants' speed. The noise at which messages were confused with each other was estimated according to the response recipient, initially stationary, ants (see Materials and Methods). Gray scale indicates the estimated overlap between every two messages $\delta(i, j)$. Note $\delta = \min(\delta(i, j)) \approx 0.3$. Data collected over $N = 278$ interactions. **d.** The mean time it takes an ant that is informed about the food to recruit two nest-mates to exit the nest is presented for two group size ranges. Error bars represent standard error of the means over $N = 24$ experiments.

are summarized below. This attempt fits in the framework of understanding what are the most restricted conditions under which broadcast of information can be achieved.

Limited message size. In the lower bound result, the size of the message is not an explicit parameter. We now instead characterize the noise by a given parameter δ that indirectly controls the size of the message (once a noise model is specified). Here, we no longer assume noise in the interactions and make the message size an explicit parameter, that ideally is as small as possible.

Memory constraint. Besides restricting the amount of bits exchanged between two agents, we also restrict the amount of memory used by agents. Importantly, the amount of memory may be larger than the message displayed by an agent. This distinction is crucial to us, and it is a major difference with the similar model of population protocols (see Section 1.6.2). We insist that agents exchange only a few bits in their messages, even if they can manipulate up to $O(\log \log n)$ bits internally.

Self-stabilization. We consider here another very hard fault tolerance constraint (on top of random interactions and limited message size). Namely, we enforce convergence for any initial internal configuration of the agents state. This is called *self-stabilization*, and it is in fact a topic of its own. In particular, imagine the source at some point switches her bit b to $1-b$. If the protocol is self-stabilizing, every agent will soon converge to $1-b$ as well, regardless of the state they were in when the source changed her mind. Such flexibility, or rather sensitivity to the changes in the source opinion, is desirable in many technological contexts, but also for biological systems. This is perhaps the biggest change with respect to the setup of Section 1.5.2.

Having defined the setting, we can now give our result.

Theorem 1.2

There exists a self-stabilizing protocol for Broadcast in the $\mathcal{PULL}$ model, that requires $O(\log n \log \log n)$ rounds, and 3 bits per message.

1.5.5 Structure of the Proofs

Let us now explain at a high level how the proof of both upper and lower bounds discussed above work, starting with the lower bound (Theorem 1.1). The idea is to reduce to a coin distinguishing task and use an information theoretic bottleneck. More precisely, if we take the perspective of a single agent, what it sees at each round is a random sample from the population, to which noise is applied. If we forget that agents adapt to what they previously saw and become more knowledgeable about the source value as time advances, it looks like it can only gain knowledge upon sampling the source, which happens with probability $\frac{1}{n}$. The task of learning the value of the source is thus tantamount to that of distinguishing between a coin with some parameter c and another one with parameter $c + \frac{1}{n}$. Standard information theoretic tools show that this requires n^2 samples. The whole proof revolves around making this intuition precise.

The proof of the upper bound is as follows. We first observe that the problem can be solved rather easily if agents have a clock (i.e., a counter), that is if they are able to have a common reference of time that is incremented on each parallel round. However, in a self-stabilizing world, it is not possible to assume this common reference, because the adversarial initialization could destroy it. We thus design a self-stabilizing clock synchronization mechanism. Our first attempt builds on a consensus protocol devised by Doerr & al [48]. It yields a clock synchronization protocol with super constant communication per message. At a high level, it consists in running the protocol of [48] on every bit of the clock (viewed as a number written in binary) *independently*. We then provide a recursive construction that improves the communication until reaching a constant number of bits. At the heart is a message reduction lemma, that crucially exploits the independence between each bit of the counter (the notion of bitwise-independence is made precise in Chapter 5).

1.6 Related Work

Some general background was already given in Section 1.3. In this section we go into further detail, and review important lines of research related to the works presented in this manuscript. The focus is on topics connecting biology and more specifically ecology and theoretical computer science. For a more complete review of interactions between biology and distributed computing, see [64].

1.6.1 Swarm Intelligence

Swarm Intelligence is a broad label designating the study of emergent collective phenomena in decentralized, agent based systems. Even if the inspiration is often taken from Nature, the algorithms derived in the context of Swarm Intelligence are usually designed to be used in a technological settings, i.e., in robotics or operations research. In those works, the behavioral observations motivate the algorithm, but the algorithm is not thought of as an accurate description of nature. It is viewed as a metaphor, useful to solve humand centered problems. This is an important difference with the works presented in this manuscript.

Ant Colony Optimization. Ant Colony Optimization, introduced by Dorigo in the early 90's is a prime example of a metaphor-based heuristic, where a situation or system found in Nature serves as a model for combinatorial search heuristic. The main underlying idea is reinforcement. When having to choose between several good candidate options (that could be housing options, routes to a food source) ants typically start by exploring all of them. Then the most promising one is signaled by some chemical mechanism. It thus attracts more individuals, reinforcing the scent marks and ultimately making it the option chosen by a strong majority, if not all individuals.

Surprisingly, the concept of reinforcement can be leveraged to derive iterative algorithms

that solve classical optimization tasks such as the traveling salesman problem[5]. Mimicking a natural entity to derive optimization algorithms is also the idea behind neural networks, or genetic algorithms. The biological entity serving as a model is the brain in the first case and Evolution in the second. It is worth noting that in the field of Ant Colony Optimization, there are usually no proofs of convergence, or mathematical guarantees on the algorithms, due in part to their sophistication. Instead we choose to study more modest algorithms, so that we are able to analyze them rigorously.

Flocking. Flocking models describe how groups of animals manage to move together, agree on a direction to follow or exhibit other group-level behavior based on simple local rules. Typically, each agent aligns its speed with that of its neighbors. Several variants exist. For instance, the word neighbors could be interpreted as the few nearest other agents, or perhaps all agents within a given distance[6]

The models are usually tested using numerical simulations, and sometimes comparisons with data. In [39], Chazelle was able to give a theoretical foundation to flocking by means of a tight bound for the flocking time. However, the bound is extremely high, which questions the relevance of the model in practice. It could also be that the worst-case assumption is overly pessimistic. On a theoretical side, flocking, as presented by Chazelle, is in fact only a particular example within the rich and technically refined field of Markov Influence Systems. In Chapter 4, we also show a lower bound for a biological phenomenon. It is however very different technically from the one of [39].

1.6.2 Population Protocols

Another important line of research to be mentioned is that of population protocols [7]. It is motivated by several biochemical contexts, such as chemical reaction networks or DNA programming. In that framework, computationally limited entities meet through pairwise interactions with the aim of computing some function of the initial state of the whole population. The first line of work on this model studied what functions can be computed under a worst case kind of scheduling [10]. It turns out to be the class of semi-linear predicates. This very interesting result says nothing however on the time bounds required for such computations. This became the focus of another line of works that emerged later [8]. The objective became to find optimal space or time bounds for explicit tasks, under a probabilistic scheduler, meaning pairs of agents are selected uniformly at random, see e.g., [3, 4, 74].

The tasks most commonly studied are *consensus*, *majority* and *leader election*. The majority problem asks to find which of two initial states A and B has the biggest count. In the leader

[5]The same idea of reinforcement also found an echo in probability theory. Reinforced random walks became an active field of research in the last decade, and the connection with ants is sometimes made explicit [111, 94].

[6] In biology, the term flocking is used only for groups of birds, whereas the term *schooling* is used for fish, and *swarming* for insect groups. Computer scientists however tend to ignore such differences, as the models for these different biological systems are similar.

election problem, each agent has the same initial state. The population should converge to having one single agent in a designated *leader* state.

1.6.3 The Beeping Model

The beeping model [2, 1], is rooted in distributed computing but has the potential to capture biological objects such as cell networks. The idea in this model is to solve a computational task over a network, such as selecting a maximal independent set under very harsh communication conditions. Namely, the communication is reduced to a single bit (a beep) per message. In a similar spirit, a central feature of the model studied in Chapter 5 is that it uses very short messages.

1.6.4 The Lower Bound Approach

Showing lower bounds is a central endeavor in computer science. The study of any computational model involves understanding its limits. Usually this means showing lower bounds on the amount of resources needed to solve a particular task. The resource can take different forms, depending on the model. Traditional computational resources are time and space but there are many others. The amount of randomness, interaction, communication can all be quantified and considered as resources as well.

An original idea introduced by Feinerman and Korman in [68] is to use the lower bound methodology to gain new knowledge on biological systems. The two-step approach is as follows. A theoretical lower bound (L) is derived in a model that captures a certain task carried by a biological entity. Then this lower bound is confronted with reality. If the system breaks the lower bound, then we infer by the contrapositive of (L) that one of the assumptions under which (L) holds is violated. This yields an indirect biological insight.

To be specific, let us give a little background on the paper [68]. It studies a collective search problem over the grid. The k agents involved in the search process cannot communicate while searching but are able to exchange some $c(k)$ bits of information prior to engaging in the search. It is shown among other things that efficient search is only possible if $c(k)$ is at least $\log\log k + \Omega(1)$. This lower bound has not yet been confronted to actual experiments. In principle, however, linking the number of searchers and the amount of time taken to find a food item should allow to form an indirect guess on how many bits of information ants exchange before performing the task under consideration.

In Chapter 4, a more refined attempt to develop this approach is presented, in a somewhat different context. This time, we do confront a theoretical lower bound about information spreading to actual experiments. Interestingly, the paper [68] was subsequently extended and improved in several ways [59, 60].

1.6.5 Approaches From Other Fields

Before biology attracted computer scientists, it has been an object of study in other fields such as ecology or physics. In those fields, it is understood that a good model should serve as a predictor, or perhaps orient the observations on the field. We share the same view as Hölldobler and Wilson [86] when they say: *Most of the predictions made from the foraging models are intuitively clear and have a double heuristic value in the study of ant ecology. First they allow a test of the basic proposition that natural selection shapes behavior [...]. Second, by framing research in terms of these models, the ecologist asks questions and searches for predicted phenomena that may otherwise easily be missed.*

Theoretical ecology There is a tradition of mathematical modeling in biology. Foraging for instance is the subject of a theory called Optimal Foraging Theory (OFT) based on economic reasoning and optimization [86, Chapter 10]. It was initiated already in the sixties [61, 78]. The topic is relevant from our perspective since all our experiments are related to group retrieval and recruitment, which take place after successful foraging.

In OFT, and more specifically the so called *central-place variant*, relevant to describe social insects living in a fixed place, the colony is viewed as a rational agent seeking to optimize a utility function, with energy being the underlying currency. A typical question that OFT seeks to address is to identify the optimal moment to leave a given high resource area, called a patch, to explore a new one. In the canonical model, the optimal moment is when the remaining resource in the patch is lower than the expected gain in other patches. Spatial modeling of the search process was traditionally not a main focal point of these works[7]. In fact, taking into account the mechanistic part of the search process, using models of intermittent search and Levy flights became a popular research direction for physicists[8]. Ecologists are aware that the actual behavior encountered in nature may not match the theoretical optimum. In some cases, some simple algorithms are studied in the form of *rules of thumbs*.

Complex systems - the physics perspective Over the last decades, physicists gradually turned their attention to biological systems. They model biological phenomena using technical tools that are common in their field. Whereas computer science mainly builds on discrete mathematics, physicists are often more comfortable using continuous objects and equations, such as PDE's. But this difference might be a superficial one.

The physics approach in fact leads to different questions as the ones we try to address. Having a computer science background, we think of any natural process we observe as an algorithm belonging to a larger family of possibilities. This naturally leads us to the following fundamental questions: in what sense the process we observe is optimal? What is the gain in a

[7] Quoting [14]: *Most models of OFT typically assume that animals have information about the location of the patches so that the time spent between patches does not come out from a search process, instead from the average distance between the patches.*

[8] In fact, the work [26] about random walks with different step lengths, presented in Section 1.7.2 is related to this line of research.

given trait or behavior with respect to another? Such questions are in fact closely tied in spirit to the theory of Evolution.

The works in this manuscript are about group retrieval by *P. longicornis* ants and recruitment in *C. niger* ants. Both phenomena are also the subject of other experiments and physics oriented works at Feinerman's group. The process of collaborative transport is studied in [75], using the Ising model, a cornerstone of statistical physics. The question investigated there is how much influence each carrier has within the group. From a single ant perspective, there is a tradeoff to be found between individuality and fully conforming to the groups effort. The local rules describing an ant behavior are connected with the macroscopic behavior of the load, which is shown to evolve at a critical regime between random walk and purely ballistic movement.

In [106], Razin & al. study the communication framework of desert ants. Amongst other things, they are able to quantify the information flow in each interaction amongst individual of this species. As explained above, this species does not communicate through pheromones but rather through physical contact, the speed of ants in some sense playing the role of an alphabet.

1.7 Works Completed During My PhD

1.7.1 Works Presented in This Document

The content of the following works on searching in the Noisy Advice model is the basis of Chapters 2 and 3. Reading Chapter 3 requires knowledge of the advice model.

[31] *Searching a Tree with Permanently Noisy Advice.*
L. Boczkowski, A. Korman, Y. Rodeh.
European Symposium on Algortihms (ESA), 2018.

[28] *Typically Fast Search on Trees with Permanently Noisy Advice.*
L. Boczkowski, U. Feige, A. Korman.
Under submission.

The following two works are presented respectively in Chapter 4 and 5.

[24, 25] *Limits for Rumor Spreading in Stochastic Populations.*
L. Boczkowski, O. Feinerman, A. Korman, E. Natale
Conference version: Innovations in Theoretical Computer Science, 2018.
Journal version: PLOS Computational Biology (ITCS), 2018.

[29, 30] *Minimizing Message Size in Stochastic Communication Patterns: Fast Self-Stabilizing Protocols with 3 bits.*
L. Boczkowski, A. Korman, E. Natale.
Conference version: Symposium on Discrete Algorithms (SODA), 2017.
Journal version: Distributed Computing, 2018.

1.7.2 Other Works

During my thesis, I had the chance to take part in other projects than the ones presented in this manuscript.

[26] *Random Walks with Multiple Step Lengths.*
L. Boczkowski, B. Guinard, A. Korman, Z. Lotker, M. Renault.
Latin American Symposium on Theoretical Informatics (LATIN), 2018.

In [26], I studied a variant of random walks where multiple step lengths are allowed. This study is in part motivated by biology, since it has been shown that the case of two different step lengths, called intermittent search, can model many animal search patterns in Nature. We analyze how long it takes such a process with k distinct step lengths, to visit every node on a cycle of a given length n. In particular, we show that if the k step lengths are chosen according to a geometric sequence, the cover time of the cycle is roughly $n^{1+O(\frac{1}{k})}$. This is shown to be tight, up to the constant hidden in the O in the exponent. Then, we try to generalize this model of search to arbitrary graphs, and show general upper bounds in that generalized variant as well.

[27] *Streaming Communication Protocols.*
L. Boczkowski, I. Kerenidis, F. Magniez.
Internationcal Colloquium on Automata, Language and Programming (ICALP), 2017.

The project [27] is about a new model combining aspects of streaming protocols and communication complexity. In the standard model of communication complexity, players are assumed to be computationally unbounded and have unrestricted access to their own input. We assume instead they have a limited memory and receive their inputs gradually, as a stream.

Both communication and memory are considered as resources to be optimized in our model. We show tradeoffs for canonical problems from the Communication Complexity literature. We also analyze the Approximate Matching problem and use it to separate the one-way variant of our model from the usual streaming model and one-way communication complexity.

Chapter 2

Searching a Tree with Noisy Local Advice

2.1 Introduction

This chapter considers a search problem on trees, in which the goal is to find a treasure that is placed at one of the nodes by an adversary. Each node of the tree holds information, called *advice*, regarding which of its neighbors is closer to the treasure, and the search may consult the advice at some nodes in order to accelerate the search.

Crucially, we assume that advice at nodes may be faulty with some probability. Many works consider noisy queries in the context of search, but it is typically assumed that queries can be resampled (see e.g., [21, 57, 62, 88]). In contrast, we assume that each location is associated with a single *permanent* advice. That is, faults are in the physical memory associated with the node, and hence querying the node again would yield the same answer. This difference is dramatic, as the search under our model does not allow for simple amplification procedures (similar to [33] albeit in the context of sorting). Searching in contexts of permanently faulty nodes has been studied in a number of works [34, 69, 79, 80, 81], but only assuming that the faulty nodes are chosen by an adversary. The difference between such worst case scenarios and the probabilistic version studied here is again significant, both in terms of results and techniques (see more details in Section 2.1.4).

2.1.1 The Noisy Advice Model

We start with some notation. Further notations are given in Section 2.1.5. Let T be an n-node tree[1] rooted at some arbitrary node σ. We consider an agent that is initially located at the root σ of T, aiming to find a node τ, called the *treasure*, which is chosen by an adversary. The

[1] We present the model for trees in this Section. As shown in the introduction of this manuscript, it can be similarly defined for arbitrary graphs (see also Section 2.7).

distance $d(u, v)$ is the number of edges on the path between u and v. The depth of a node u is $d(u) = d(\sigma, u)$. Let $d = d(\tau)$ denote the depth of τ, and let the depth D of the tree be the maximal depth of a node. Finally, let Δ_u denote the degree of node u and let Δ denote the maximal degree in the tree.

Each node $u \neq \tau$ is assumed to be provided with an *advice*, termed $\texttt{adv}(u)$, which provides information regarding the direction of the treasure. Specifically, $\texttt{adv}(u)$ is a pointer to one of u's neighbors. It is called *correct* if the pointed neighbor is one step closer to the treasure than u is. Each vertex $u \neq \tau$ is *faulty* with probability q (the meaning of being faulty will soon be explained). Otherwise, u is considered *sound*, in which case its advice is correct. We call q the *noise parameter*. Unless otherwise stated, this parameter is the same across all nodes, but in some occasions, we also allow it to vary across nodes. In that case q is defined as $\max_u(q_u)$.

Random and semi-adversarial variant. We consider two models for faulty nodes. The main model assumes that the advice at a faulty node points to one of its neighbors chosen uniformly at random (and so possibly pointing at the correct one). We also consider an adversarial variant, called the *semi-adversarial model*, where this neighbor is chosen by an oblivious adversary. That is, an adversary specifies for each node what advice it would have assuming it is faulty. Then, faulty nodes are still chosen randomly as in the main model, but their advice is specified by the adversary.

Move and query complexity. The agent can move by traversing edges of the tree. At any time, the agent can query its hosting node in order to "see" the corresponding advice and to detect whether the treasure is present there. The protocol terminates when the agent queries the treasure. We evaluate a search algorithm A by two measures: The move complexity is the number of edge traversals. It is the focus of this chapter. We are also interested in the query complexity, which is the number of times advice needs to be probed. This chapter focuses on counting moves, but query complexity is convenient to phrase lower bounds. The number of queries is always smaller than the number of moves.

Noise assumption. The noise parameter q governs the accuracy of the environment. If $q = 0$ for all nodes, then advice is always correct. This case allows to find the treasure in d moves, by simply following each encountered advice. On the other extreme, if $q = 1$, then advice is essentially meaningless, and the search cannot be expected to be efficient. An intriguing question is therefore to identify the largest value of q that allows for efficient search.

Expectation and high probability. Importantly, we consider two kinds of guarantees: expectation and high probability. In the first case, we measure the performance of the algorithm as the expected number of moves before the treasure is found. By default, expectation is taken over both the randomness involved in sampling advice and the possible probabilistic choices made by A. In the second case, we only want to find a constant probability event under which

the treasure is found fast. Interestingly, the two settings lead to quite different thresholds and techniques.

A bound on expectation, can be converted into a high probability statement through the Markov inequality. It happens that we derive a result in expectation by first gaining control of the algorithm under a large probability event, and then taming the dangerous low probability event where problems occur. These two facts indicate that high probability guarantees tend to be easier to achieve than expectation ones. On the other hand, the high probability lower bound presented in this chapter is arguably harder than its expectation counterpart.

2.1.2 Results in Expectation

We start by presenting our results for the average case. Consider the noisy advice model on trees with maximum degree Δ and depth D. Roughly speaking, we show that $1/\sqrt{\Delta}$ is the threshold for the noise parameter q, in order to obtain search algorithms with low expected complexities.

The proof that there is no algorithm with low expected complexities when the noise exceeds $1/\sqrt{\Delta}$ is rather simple, and in fact, holds even if the algorithm has access to the advice of all internal nodes. Intuitively, the argument is as follows (the formal proof appears in Section 2.3). Consider a complete Δ-ary tree of depth D and assume that the treasure τ is placed at a leaf. The first observation (Lemma 2.10) is that the expected number of leaves having more advice point to them than to τ is a lower bound on the query complexity. The next observation is that there are roughly Δ^D leaves whose distance from τ is $2D$. For each of those leaves u, the probability that more advice points towards it than towards τ can be approximated by the probability that all nodes on path connecting u and τ are faulty. As this latter probability is q^{2D}, the expected number of leaves that have more pointers leading to them is roughly $\Delta^D q^{2D}$, which explodes when $q \gg 1/\sqrt{\Delta}$. This essentially establishes the lower bound for the noise regime.

In Section 2.2, we present an algorithm that is optimal up to a constant factor for the regime of noise below the threshold. Furthermore, this algorithm does not require prior knowledge of either the tree's structure, or the values of Δ, q, d, or n.

In this section, we extend slightly the model, by allowing each node v to have a distinct mistake parameter q_v. This greater flexibility makes our results stronger. It also happens to be convenient from a technical standpoint. The following technical definition is used in our results, in place of the more crude $q < \frac{1}{\sqrt{\Delta}}$ given in Table 2.1.

Definition 2.1

Condition ($\star$) holds with parameter $0 < \varepsilon < 1$ if for every node v, we have

$$q_v < \frac{1 - \varepsilon - \Delta_v^{-\frac{1}{4}}}{\sqrt{\Delta}_v + \Delta_v^{\frac{1}{4}}}.$$

Since $\Delta_v \geq 2$, the condition is always satisfiable when taking a small enough ε. In the following theorems the $\mathcal{O}$ notation hides only a polynomial a polynomial term in $1/\varepsilon$.

All our algorithms are deterministic, hence, expectation is taken with respect only to the sampling of the advice.

Theorem 2.2

For any $\varepsilon > 0$, if Condition $(\star)$ holds with parameter ε there exists a deterministic walking algorithm $\mathtt{A}_{walk}$ that requires $\mathcal{O}(\sqrt{\Delta}d)$ moves in expectation.

Lower Bounds in Expectation. We establish in Section 2.3 the following lower bounds.

Theorem 2.3

For any randomized algorithm A and any integer $\Delta \geq 3$, we have

1. **Exponential complexity above the threshold.**
   Consider a complete Δ-ary tree. For every constant $\varepsilon > 0$, if $q \geq \frac{1+\varepsilon}{\sqrt{\Delta-1}} \cdot (1 + \frac{1}{\Delta-1})$, then A move complexity in expectation is exponential in D.
2. For any integer d, there is a tree with at most $d\Delta$ nodes, and a placement of the treasure at depth d, such that A requires $\Omega(dq\Delta)$ moves in expectation.

Observe that taken together, Theorems 2.2,2.3 and Condition $(\star)$ imply that for any $\varepsilon > 0$ and large enough Δ, efficient search can be achieved if $q < (1-\varepsilon)/\sqrt{\Delta}$ but not if $q > (1+\varepsilon)/\sqrt{\Delta}$.

Memoryless Algorithms. Finally, we analyze the performance of simple memoryless algorithms called *probabilistic following*, suggested in [71]. At every step, the algorithm follows the advice at the current vertex with some fixed probability λ, and performs a random walk step otherwise. It turns out that such algorithms can perform well, but only in a very limited regime of noise. Specifically, we prove:

Theorem 2.4

There exist positive constants c_1, c_2 and c_3 such that the following holds. If for every vertex u, $q_u < c_1/\Delta_u$ then there exists a probabilistic following algorithm that finds the treasure in less than $c_2 d$ expected steps. On the other hand, if $q > c_3/\Delta$ then for any probabilistic following strategy the move complexity on a complete Δ-ary tree is exponential in the depth of the tree.

Since this algorithm is randomized, expectation is taken over both the randomness involved in sampling advice and the possible probabilistic choices made by the algorithm.

Interestingly, when $q_u < c_1/\Delta_u$ for all vertices, this algorithm works even in a semi-adversarial model. In fact, it turns out that in the semi-adversarial model, probabilistic following algorithms are the best possible, as the threshold for efficient search, with respect to any algorithm, is roughly $1/\Delta$. There results are discussed and proved in Section 2.4.

2.1.3 Results in High Probability

We start with the following upper bound. The proof is presented in Section 2.5.

Theorem 2.5

Let $0 < \varepsilon < 1/2$ be a constant, and suppose that $q = \Delta^{-\varepsilon}$, and that Δ is sufficiently large ($\Delta \geq 2^{6/\varepsilon^2}$ suffices). Then there exists a moving algorithm A_{walk} that discovers τ in $(\frac{d}{\delta})^{O(\frac{1}{\varepsilon})}$ moves with probability $1-\delta$. Moreover, the statement holds even in the semi-adversarial variant.

The upper bounds shown in Theorem 2.5 is matched up to the constant in the exponent, by the following lower bound, presented in Section 2.6.

Theorem 2.6

Let $0 < \varepsilon < 1/2$ be an arbitrary constant, and suppose that $q \leq \Delta^{-\varepsilon}$, and that D is sufficiently large, as a function of Δ and ε. On the complete Δ-ary tree of depth D, any algorithm with success probability $1-\delta$ needs at least $(\delta^{-1}D)^{\frac{1-\varepsilon}{\varepsilon}(1+o_D(*))}$ queries (and consequently also moves) before finding τ with constant probability. ($o_D(\cdot)$ denotes a function of D that tends to 0 as D grows.) The statement also holds with respect to randomized algorithms.

2.1.4 Related Work

In computer science, search algorithms have been the focus of numerous works. Due to their importance, trees are particularly popular structures to investigate search, see e.g., [92, 19, 103, 101]. Within the literature on search, many works considered noisy queries [62, 88, 57], however, it was typically assumed that noise can be *resampled* at every query. As mentioned, dealing with permanent faults incurs challenges that are fundamentally different from those that arise when allowing queries to be resampled. To illustrate this difference, consider the simple example of a star graph and a constant $q < 1$. Straightforward amplification can detect the target in $\mathcal{O}(1)$ expected number of queries. In contrast, in our model, it can be easily seen that $\Omega(n)$ is a lower bound for both the move and the query complexities, for any constant noise parameter.

A search problem on graphs in which the set of nodes with misleading advice is chosen by an adversary was studied in [79, 80, 81], as part of the more general framework of the *liar*

models [11, 32, 41, 104]. Data structures with adversarial memory faults have been investigated in the so called faulty-memory RAM model introduced in [70]. In particular, data structures supporting the same operations as search trees with adversarial memory faults were studied in [69, 34]. Interestingly, the data structures developed in [34] can cope with up to $O(\log n)$ faults, happening at any time during the execution of the algorithm, while maintaining optimal space and time complexity. It is important to observe that all these models take worst case assumptions, leading to technical approaches and results which are very different from what one would expect in average-case analysis. Persistent probabilistic memory faults, as we study here, have been explicitly studied in [33], in the context of sorting. Persistent probabilistic errors were also studied in contexts of learning and optimization, see [82].

The noisy advice model considered in this chapter actually originated in the recent biologically centered work [71], aiming to abstract navigation relying on guiding instructions in the context of collaborative transport by ants. In that work, the authors modeled ant navigation as a probabilistic following algorithm, and noticed that an execution of such an algorithm can be viewed as an instance of Random Walks in Random Environment (RWRE) [112, 53]. Relying on results from this subfield of probability theory, the authors showed that when tuned properly, such algorithms enjoy linear move complexity on grids, provided that the bias towards the correct direction is sufficiently high.

2.1.5 Notations

Here we give some notation used throughout the chapter, leaving aside the terminology that is specific to a given part or result.

Denote $p = 1 - q$, and for a node u, $p_u = 1 - q_u$. For two nodes u, v, let $\langle u, v\rangle$ denote the simple path connecting them, excluding the end nodes, and let $[u, v\rangle = \langle u, v\rangle \cup \{u\}$ and $[u, v] = [u, v\rangle \cup \{v\}$. For a node u, let $T(u)$ be the subtree rooted at u. We denote by $\overrightarrow{\texttt{adv}}(u)$ (resp. $\overleftarrow{\texttt{adv}}(u)$) the set of nodes whose advice points towards (resp. away from) u. By convention $u \notin \overrightarrow{\texttt{adv}}(u) \cup \overleftarrow{\texttt{adv}}(u)$. Unless stated otherwise, log is the natural logarithm.

The nodes on the path from the root σ to the treasure τ are named as $[\sigma, \tau] := \{v_0 = \sigma, v_1, \ldots, v_{d-1}, v_d = \tau\}$. We say that node v is a *descendant* of node u if u lies on the path from σ to v, and v is a *child* of u if it is both a descendant of u and a neighbor of u.

2.1.6 Organization of This Chapter

In Section 2.2 we present our optimal walking algorithm in expectation. The associated lower bound is the focus of Section 2.3. Theorem 2.4 and the threshold of $\Theta(1/\Delta)$ that applies to the semi-adversarial setting are proved in Section 2.4. Section 2.5 is devoted to the high probability algorithm mentioned in Theorem 2.5, while Section 2.6 is about the corresponding lower bound. In Section 2.7, we give a list of open problems.

Table 2.1 summarizes the results presented in this chapter, in a simplified form.

	Upper Bound		Lower Bound	
	Regime	Moves	Regime	Moves
Expectation	$q \ll \frac{1}{\sqrt{\Delta}}$	$\mathcal{O}(d\sqrt{\Delta})$	$q \gg \frac{1}{\sqrt{\Delta}}$	$e^{\Omega(d)}$
High Probability	$q = \Delta^{-\varepsilon}$	$d^{O(\varepsilon^{-1})}$	$q = \Delta^{-\varepsilon}$	$d^{\Omega(\varepsilon^{-1})}$

Figure 2.1: A summary of the results presented in this chapter, in simplified form. The precise conditions behind the symbol $\ll$ will be clarified later.

2.2 Optimal Walking Algorithm in Expectation

In this section we prove Theorem 2.2. At a high level, at any given time, the walking algorithm processes the advice seen so far, identifies a promising node to continue from on the border of the already discovered connected component, moves to that node, and explores one of its neighbors. The crux of the matter is identifying the correct formalisation of *promising*, that leads to an efficient algorithm. This, we will see, amounts to finding a correct *prior* for the treasure location.

2.2.1 Algorithm Design following a Greedy Bayesian Approach

In our setting the treasure is placed by an adversary. However, we can still study algorithms induced by assuming that it is placed according to some known distribution and see how they measure up in our worst case setting. As mentioned, this approach is similar to [21], which studies the closely related, yet much simpler problem of search on the line. Of course, the success of this scheme highly depends on the choice of the prior distribution we take.

To make our life easier, let us first assume that the structure of the tree is known to the algorithm. Also, we assume the treasure is placed according to some known distribution θ supported on the leaves, and denote by `adv` the advice on the nodes we have already visited. Aiming to find the treasure as fast as possible, a possible greedy algorithm explores the vertex that, given the advice seen so far, has the highest probability of having the treasure in its subtree.

We extend the definition of θ to internal nodes by defining $\theta(u)$ to be the sum of $\theta(w)$ over all leaves w of $T(u)$. Given some u that was not visited yet, and given the previously seen

advice $\texttt{adv}$, the probability of the treasure being in u's subtree $T(u)$, is:

$$\begin{aligned}\mathbb{P}\left(\tau \in T(u) \,|\, \texttt{adv}\right) &= \frac{\mathbb{P}\left(\tau \in T(u)\right)}{\mathbb{P}\left(\texttt{adv}\right)}\mathbb{P}\left(\texttt{adv} \,|\, \tau \in T(u)\right) \\ &= \frac{\theta(u)}{\mathbb{P}\left(\texttt{adv}\right)} \prod_{w \in \overrightarrow{\texttt{adv}}(u)} \left(p_w + \frac{q_w}{\Delta_w}\right) \prod_{w \in \overleftarrow{\texttt{adv}}(u)} \frac{q_w}{\Delta_w}.\end{aligned}$$

The last factor is q_w/Δ_w because it is the probability that the advice at w points exactly the way it does in $\texttt{adv}$, and not only away from τ. Note that the advice seen so far is never for vertices in $T(u)$ as we consider a walking algorithm, and u has not been visited yet. Therefore, if $\tau \in T(u)$ then correct advice in $\texttt{adv}$ points to u. We ignore the term $p_w + q_w/\Delta_w$ as it is normally quite close to 1, and applying a log we can approximate the relative strength of a node by:

$$\log(\theta(u)) + \sum_{w \in \overleftarrow{\texttt{adv}}(u)} \log\left(\frac{q_w}{\Delta_w}\right).$$

We do not want to assume that our algorithm knows q_w, but we do assume that in the worst scenario $q_w \sim 1/\sqrt{\Delta_w}$. Assigning this value and rescaling we finally define:

$$\texttt{score}(u) = \frac{2}{3}\log(\theta(u)) - \sum_{w \in \overleftarrow{\texttt{adv}}(u)} \log(\Delta_w).$$

When comparing two specific vertices u and v, $\texttt{score}(u) > \texttt{score}(v)$ iff:

$$\sum_{w \in \langle u,v\rangle \cap \overrightarrow{\texttt{adv}}(u)} \log(\Delta_w) - \sum_{w \in \langle u,v\rangle \cap \overrightarrow{\texttt{adv}}(v)} \log(\Delta_w) > \frac{2}{3}\log\left(\frac{\theta(v)}{\theta(u)}\right).$$

This is because any advice that is not on the path between u and v contributes the same to both sides, as well as advice on vertices on the path that point sideways, and not towards u or v[2]. Since we use this score to compare two vertices that are neighbors of already explored vertices, and our algorithm is a walking algorithm, then we will always have all the advice on this path. In particular, the answer to whether $\texttt{score}(u) > \texttt{score}(v)$, does not depend on the specific choices of the algorithm, and does not change throughout the execution of the algorithm, even though the scores themselves do change. The comparison depends only on the advice given by the environment.

Let us try and justify the score criterion at an intuitive level. Consider the case of a complete Δ-ary tree, with θ being the uniform distribution on the leaves[3]. Here $score(u) > score(v)$ if

[2] It is tempting to define $\texttt{score}(u)$ as the sum of weighted advice from the root to u. However, when comparing two vertices, the advice of their least common ancestor would be counted twice, which we prefer to avoid.

[3] Actually, a similar formula could be derived choosing θ to be the uniform distribution over all nodes, but for technical reasons it is easier to restrict it to leaves only.

(cheating a little by thinking of $\log(\Delta)$ and $\log(\Delta-1)$ as equal):

$$\left|\overrightarrow{\texttt{adv}}(u)\cap\langle u,v\rangle\right| - \left|\overrightarrow{\texttt{adv}}(v)\cap\langle u,v\rangle\right| > \frac{2}{3}\big(d(u)-d(v)\big).$$

If, for example, we consider two vertices $u, v \in T$ at the same depth, then $score(u) > score(v)$ if there is more advice pointing towards u than towards v. If the vertices have different depths, then the one closer to the root has some advantage, but it can still be beaten.

For general trees, perhaps the most natural θ is the uniform distribution on all nodes (or just on all leaves - this choice is actually similar). It is also a generalization of the example above. Unfortunately, however, while this works well on the complete Δ-ary tree, this approach fails on other (non-complete) Δ-ary trees (see the full version of this work [31] for details).

2.2.2 Algorithm $\texttt{A}_{walk}$

In our context, there is no distribution over treasure location and we are free to choose θ as we like. We take θ to be the distribution defined by a simple random process. Starting at the root, at each step, walk down to a child uniformly at random. until reaching a leaf. For a leaf v, define $\theta(v)$ as the probability that this process eventually reaches v. Our extension of θ can be interpreted as $\theta(v)$ being the probability that this process passes through v. Formally, $\theta(\sigma)=1$, and $\theta(u) = (\Delta_\sigma \prod_{w\in\langle\sigma,u\rangle}(\Delta_w-1))^{-1}$. It turns out that this choice, slightly changed, works remarkably well, and gives an optimal algorithm in noise conditions that practically match those of our lower bound. For a vertex $u \neq \sigma$, define:

$$\beta(u) = \prod_{w\in[\sigma,u\rangle} \Delta_w.$$

It is a sort of approximation of $1/\theta(u)$, which we prefer for technical convenience. Indeed, for all u, $1/\beta(u) \leq \theta(u)$. A wonderful property of this β (besides the fact that it gives rise to an optimal algorithm) is that to calculate $\beta(v)$ (just like θ), one only needs to know the degrees of the vertices from v up to the root. It is hard to imagine distributions on leaves that allow us to calculate the probability of being in a subtree without knowing anything about it!

In the walking algorithm, if v is a candidate for exploration, these nodes must have been visited already and so the algorithm does not need any a priori knowledge of the structure of the tree. The following claim will be soon useful:

Claim 2.7

The following two inequalities hold for every $c < 1$:

$$\sum_{v\in T}\frac{c^{d(v)}}{\beta(v)} \leq \frac{1}{1-c}, \quad \sum_{v\in T}\frac{d(v)c^{d(v)}}{\beta(v)} \leq \frac{c}{(1-c)^2}.$$

Proof. To prove the first inequality, follow the same random walk defining θ. Starting at the root, at each step choose uniformly at random one of the children of the current vertex. Now, while passing through a vertex v collect $c^{d(v)}$ points. No matter what choices are made, the number of points is at most $1 + c + c^2 + ... = 1/(1-c)$. On the other hand, $\sum_{v \in T} \theta(v) c^{d(v)}$ is the expected number of points gained. The result follows since $1/\beta(v) \leq \theta(v)$. The second inequality is derived similarly, using the fact that $c + 2c^2 + 3c^3 + \ldots = c/(1-c)^2$. □

For a vertex $u \in T$ and previously seen advice `adv` define:

$$\texttt{score}(u) = \frac{2}{3} \log\left(\frac{1}{\beta(u)}\right) - \sum_{w \in \overleftarrow{\texttt{adv}}(u)} \log(\Delta_w).$$

Algorithm $\texttt{A}_{walk}$ keeps track of all vertices that are children of the vertices it explored so far, and repeatedly walks to and then explores the one with highest score according to the current advice, breaking ties arbitrarily. As stated in the introduction, the algorithm does not require prior knowledge of either the tree's structure, or the values of Δ, q, d or n.

2.2.3 Analysis

Recall the definition of Condition ($\star$) from Definition 2.1. The next lemma provides a large deviation bound tailored to our setting.

Lemma 2.8

Consider independent random variables $X_1, \ldots, X_\ell$, where X_i takes the values $(-\log \Delta_i, 0, \log \Delta_i)$ with respective probabilities $(p_i + \frac{q_i}{\Delta_i}, q_i(1 - \frac{2}{\Delta_i}), \frac{q_i}{\Delta_i})$, for parameters $p_i, q_i = 1 - p_i$ and $\Delta_i > 0$. Assume that Condition ($\star$) holds for some $\varepsilon > 0$. Then for every integer (positive or negative) m,

$$\mathbb{P}\left(\sum_{i=1}^{\ell} X_i \geq m\right) \leq \frac{(1-\varepsilon)^\ell}{e^{\frac{3m}{4}}} \prod_{i=1}^{\ell} \frac{1}{\sqrt{\Delta_i}}.$$

Proof. For any $s \in \mathbb{R}$,

$$\begin{aligned}
\mathbb{P}\left(\sum_{i=1}^{\ell} X_i \geq m\right) &= \mathbb{P}\left(e^{s\sum_{i=1}^{\ell} X_i} \geq e^{sm}\right) \leq \frac{\mathbb{E}\left[e^{s\sum_i X_i}\right]}{e^{sm}} = \frac{\prod_i \mathbb{E}\left[e^{sX_i}\right]}{e^{sm}} \\
&= \frac{1}{e^{sm}} \prod_{i=1}^{\ell} \left(\frac{p_i + \frac{q_i}{\Delta_i}}{e^{\log(\Delta_i)s}} + q_i\left(1 - \frac{2}{\Delta_i}\right) + \frac{q_i}{\Delta_i} e^{\log(\Delta_i)s}\right) \\
&\leq \frac{1}{e^{sm}} \prod_{i=1}^{\ell} \left(\frac{1}{\Delta_i^s} + q_i + q_i \Delta_i^{s-1}\right).
\end{aligned}$$

We take $s = \frac{3}{4}$, and get:

$$\mathbb{P}\left(\sum_{i=1}^{\ell} X_i \geq m\right) \leq \frac{1}{e^{\frac{3m}{4}}} \prod_{i=1}^{\ell} \left(\Delta_i^{-\frac{3}{4}} + q_i + q_i \Delta_i^{-\frac{1}{4}}\right) \leq \frac{1}{e^{\frac{3m}{4}}} \prod_{i=1}^{\ell} \frac{1-\varepsilon}{\sqrt{\Delta_i}}$$

Where for the last step we used Condition ($\star$) which says:

$$q_i < \frac{1-\varepsilon-\Delta_i^{-\frac{1}{4}}}{\sqrt{\Delta_i}+\Delta_i^{\frac{1}{4}}} q_i \Delta_i^{\frac{1}{2}} + q_i \Delta_i^{\frac{1}{4}} + \Delta_i^{-\frac{1}{4}} < 1-\varepsilon$$

$$\Delta_i^{-\frac{3}{4}} + q_i + q_i \Delta_i^{-\frac{1}{4}} < \frac{1-\varepsilon}{\sqrt{\Delta_i}}$$

□

The next theorem states that $\mathtt{A}_{walk}$ is optimal up to a constant factor for the regime of noise below the threshold. It establishes Theorem 2.2.

Theorem 2.9

Assume that Condition ($\star$) holds for some fixed $\varepsilon > 0$. Then $\mathtt{A}_{walk}$ requires only $\mathcal{O}(d\sqrt{\Delta})$ moves in expectation. The constant hidden in the $\mathcal{O}$ notation only depends polynomially on $1/\varepsilon$.

Proof. Denote the vertices on the path from σ to τ by $\sigma = u_0, u_1, \ldots, u_d = \tau$ in order. Denote by E_k the expected time to reach u_k once u_{k-1} is reached. We will show that for all k, $E_k = \mathcal{O}(\sqrt{\Delta})$, and by linearity of expectation this concludes the proof.

Once u_{k-1} is visited, $\mathtt{A}_{walk}$ only goes to some of the nodes that have score at least as high as u_k. We can therefore bound E_k from above by assuming we go through all of them, and this expression does not depend on the previous choices of the algorithm and the nodes it saw before seeing u_k. The length of this tour is bounded by twice the sum of distances of these nodes from u_k. Hence,

$$E_k \leq 2 \sum_{i=1}^{k} \sum_{u \in C(u_i)} \mathbb{P}\left(\mathtt{score}(u) \geq \mathtt{score}(u_k)\right) \cdot d(u_k, u).$$

Where $C(u_k) = T(u_{k-1}) \setminus T(u_k)$, and so $\cup_{i=1}^{k} C(u_i) = T \setminus T(u_k)$. Recall that scores are defined so that u has a larger score than u_k, if the sum of weighted arrows on the path $\langle u_k, u \rangle$ is at least $\frac{2}{3}\log(\beta(u)/\beta(u_k))$. Setting m to be this value, Lemma 2.8 allows to calculate this probability exactly. Indeed, a vertex x on the path should point towards u_k: this happens with probability $p_x + q_x/\Delta_x$. Otherwise, it points towards u with probability q_x/Δ_x, and elsewhere

with probability $q_x(1-2/\Delta_x)$. Denoting $c = 1-\varepsilon$,

$$\begin{aligned}\frac{E_k}{2} &\le \sum_{i=1}^{k} \sum_{u\in C(u_i)} \frac{c^{d(u_k,u)-1}}{e^{\frac{3}{4}\cdot\frac{2}{3}\log\left(\frac{\beta(u)}{\beta(u_k)}\right)}} \sqrt{\prod_{v\in\langle u,u_k\rangle} \frac{1}{\Delta_v}} \cdot d(u_k,u) \\ &= \frac{1}{c}\sum_{i=1}^{k} \sum_{u\in C(u_i)} \frac{c^{d(u_k,u)}}{\sqrt{\frac{\beta(u)}{\beta(u_k)}}} \sqrt{\frac{\Delta_{u_i}}{\frac{\beta(u_k)}{\beta(u_i)}\cdot\frac{\beta(u)}{\beta(u_i)}}} \cdot d(u_k,u) \\ &\le \frac{\sqrt{\Delta}}{c}\sum_{i=1}^{k} c^{d(u_k,u_i)} \sum_{u\in C(u_i)} c^{d(u_i,u)}\frac{\beta(u_i)}{\beta(u)} \cdot \Big(d(u_k,u_i)+d(u_i,u)\Big).\end{aligned}$$

By Claim 2.7, applied to the tree rooted at u_i, we get:

$$\sum_{u\in C(u_i)} \frac{c^{d(u_i,u)}\beta(u_i)}{\beta(u)} < \frac{1}{1-c}, \quad \text{and} \quad \sum_{u\in C(u_i)} \frac{c^{d(u_i,u)}\beta(u_i)}{\beta(u)} d(u_i,u) < \frac{c}{(1-c)^2}.$$

And so:

$$\begin{aligned}\frac{E_k}{2} &\le \frac{\sqrt{\Delta}}{c(1-c)}\sum_{i=1}^{k} c^{d(u_k,u_i)} d(u_k,u_i) + \frac{\sqrt{\Delta}}{(1-c)^2}\sum_{i=1}^{k} c^{d(u_k,u_i)} \\ &\le \frac{(1+c)\sqrt{\Delta}}{(1-c)^3} \le \frac{2\sqrt{\Delta}}{\varepsilon^3} = \mathcal{O}\left(\sqrt{\Delta}\right),\end{aligned}$$

where we again used the equality $c + 2c^2 + 3c^3 + \ldots = c/(1-c)^2$. □

2.3 Lower bounds in Expectation

2.3.1 Exponential Complexity Above the Threshold

We wish to prove Item (1) in Theorem 2.3. Namely, that for every fixed $\varepsilon > 0$, and for every complete Δ-ary tree, if $q \ge \frac{1+\varepsilon}{\sqrt{\Delta-1}} \cdot (1 + \frac{1}{\Delta-1})$, then every randomized search algorithm has query (and move) complexity which is both exponential in the depth d of the treasure and polynomial in n. In fact, this lower bound holds even if the algorithm has access to the advice of all internal nodes. The following lemma is proved in below, in Section 2.3.2:

Lemma 2.10

Assume the treasure is placed in a leaf τ of the complete Δ-ary tree. Denote by $\mathtt{adv}$ the random advice on all internal nodes, then the expected number of leaves u satisfying $|\overrightarrow{\mathtt{adv}}(u)| > |\overrightarrow{\mathtt{adv}}(\tau)|$, is a lower bound on the query complexity of any algorithm.

Using Lemma 2.10, all we need to do is approximate the number of leaves u satisfying $|\overrightarrow{\texttt{adv}}(u)| > |\overrightarrow{\texttt{adv}}(\tau)|$. When comparing the number of pointers that point towards each of two different nodes, only the pointers of the internal nodes on the path between them may influence on the result. The probability that a leaf u "beats" the treasure τ in the sense of Lemma 2.10, is at least the probability that exactly one node on the path points to u and none of the rest point towards the treasure. This probability is at least

$$\frac{q}{\Delta} \cdot \left(q \cdot \left(1 - \frac{1}{\Delta} \right) \right)^{d(u,\tau)-2}.$$

There are precisely $(\Delta - 1)^D$ leaves whose distance from the treasure is $2D$. Therefore, the expected number of leaves that beat the treasure is at least:

$$\begin{aligned} \frac{q}{\Delta}(\Delta-1)^D q^{2D-2} \cdot \left(1 - \frac{1}{\Delta}\right)^{2D-2} &= \frac{\Delta}{q(\Delta-1)^2} \cdot \left(\frac{q^2(\Delta-1)^3}{\Delta^2} \right)^D \\ &\geq \frac{\Delta}{q(\Delta-1)^2} \cdot (1+\varepsilon)^2 D. \end{aligned}$$

Item (1) in Theorem 2.3 follows.

2.3.2 Proof of Lemma 2.10

For the lower bound, assume the algorithm is given the advice $\texttt{adv}$ for all the internal nodes for free. By Yao's principle, instead of taking the worst case placement of the treasure for a randomized algorithm, we obtain a lower bound by considering only deterministic algorithms when the treasure is placed uniformly at random at one of the leaves.

In this simplified setting, an optimal algorithm can be described explicitly: It sorts the leaves according $\mathbb{P}(\cdot \,|\, \texttt{adv})$ (Claim 2.11) and tries them in this order. This order in fact corresponds to ranking nodes by how many arrows point to them (Claim 2.12). The expected number of nodes which are higher than the treasure in this ordering is therefore a lower bound for this algorithm, and thus for all algorithms.

Let $\mathcal{L}$ be the set of leaves. For a given leaf $u \in \mathcal{L}$ and an advice configuration $\texttt{adv}$, let $C(\texttt{A}, \texttt{adv}, u)$ be the cost (number of queries) of $\texttt{A}$ when the advice is equal to $\texttt{adv}$ and the treasure is located at u. We also define the cost $C(\texttt{A}, u)$ of an algorithm $\texttt{A}$ when the treasure τ is located at u to be the expected cost of $\texttt{A}$ before finding τ where the expectation is over advice setting. That is:

$$C(\texttt{A}, u) = \sum_{\texttt{adv}} C(\texttt{A}, \texttt{adv}, u) \mathbb{P}(\texttt{adv} \,|\, u).$$

In our setting, the expected number of queries of $\texttt{A}$ is:

$$C(\texttt{A}) = \sum_{u \in \mathcal{L}} \mathbb{P}(u) \sum_{\texttt{adv}} C(\texttt{A}, \texttt{adv}, u) \mathbb{P}(\texttt{adv} \,|\, u).$$

Claim 2.11

The algorithm $\mathtt{A}$ that tries the locations u in the order given by $\mathbb{P}(u \mid \mathtt{adv})$, i.e., the most likely u is tried first and the least likely tried last, minimizes $C(\mathtt{A})$.

Proof. We can write

$$C(\mathtt{A}) = \sum_{\mathtt{adv}} \mathbb{P}(\mathtt{adv}) \sum_{u \in \mathcal{L}} C(\mathtt{A}, \mathtt{adv}, u) \mathbb{P}(u \mid \mathtt{adv}),$$

where it is understood that $\mathbb{P}(\mathtt{adv})$ is the marginal of $\mathbb{P}(\mathtt{adv}, u)$ with respect to the advice. The term $\mathbb{P}(u \mid \mathtt{adv})$, standing for the probability of u holding the treasure given that the advice configuration is $\mathtt{adv}$, is only defined because we assume the treasure is placed according to a known distribution (uniform in our case). For a fixed advice setting $\mathtt{adv}$, it follows from the *rearrangement inequality* that $\sum_{u \in \mathcal{L}} C(\mathtt{A}, \mathtt{adv}, u) \mathbb{P}(u \mid \mathtt{adv})$ is minimized when $C(\mathtt{A}, \mathtt{adv}, u)$ and $\mathbb{P}(u \mid \mathtt{adv})$ are sorted in the same order with respect to u. This corresponds to algorithm $\mathtt{A}$ trying the locations u in the order given by $\mathbb{P}(u \mid \mathtt{adv})$, which is exactly the statement of the claim. Hence, since we assume that all advice is known, the algorithm we have just described is feasible, and, in fact, optimal. Moreover, its query complexity is at least 1 plus the expected number of nodes which are strictly more likely than the treasure, where the expectation is taken over the randomness of the advice. □

It only remains to check that a node u is more likely than τ given an advice setting $\mathtt{adv}$ iff more arrows point to u than τ. This will conclude the proof of Lemma 2.10 and hence of the exponential lower bound in Theorem 2.3.

Claim 2.12

For two leaves $u, v \in \mathcal{L}$, and advice configuration $\mathtt{adv}$, $\mathbb{P}(u \mid \mathtt{adv}) > \mathbb{P}(v \mid \mathtt{adv})$ if and only if there is more advice pointing towards u than advice pointing towards v.

Proof. Recall that, by definition of the model

$$\mathbb{P}(\mathtt{adv} \mid \tau = u) = \left(p + \frac{q}{\Delta}\right)^{|\overrightarrow{\mathtt{adv}}(u)|} \left(q(1 - \frac{1}{\Delta})\right)^{|\overleftarrow{\mathtt{adv}}(u)|},$$

In our regime it will always be the case that $p + \frac{q}{\Delta} > q(1 - \frac{1}{\Delta})$, simply because we assume $q < p$. Hence $\mathbb{P}(\mathtt{adv} \mid \tau = u)$ is an increasing function of $|\overrightarrow{\mathtt{adv}}(u)|$.

Since τ is placed uniformly at random, it follows from Bayes rule that $\mathbb{P}(\mathtt{adv} \mid \tau = u) \propto \mathbb{P}(\tau = u \mid \mathtt{adv})$. The symbol $\propto$ indicates that we omit the renormalizing factor. Hence, we obtain that $\mathbb{P}(\tau = u \mid \mathtt{adv}) > \mathbb{P}(\tau = v \mid \mathtt{adv})$ if and only if $|\overrightarrow{\mathtt{adv}}(u)| > |\overrightarrow{\mathtt{adv}}(v)|$. □

2.3.3 A Lower Bound for the Move Complexity in Expectation Below the Threshold

Observation 2.13

For any Δ and d, there exists a tree of depth d and maximal degree at most Δ for which any search algorithm A requires at least $\Omega(dq\Delta)$ moves in expectation.

Proof. To see why the observation holds consider the caterpillar tree, composed of a path of length n/Δ with each of its nodes being the center of a star graph of degree Δ. Assume that the agent starts at one of the end sides of the path and the treasure at distance d on the caterpillar spine. Recall that we assume that the algorithm does not know the tree structure. in expectation, $\Omega(dq)$ nodes will point at an incorrect neighbor, and to pass from any of those to the next node on the path, will require the agent to perform $\Omega(\Delta)$ trials in expectation. □

2.4 Memoryless Algorithms and the Semi-Adversarial Model in Expectation

In this section we present our results on the memoryless algorithms described in the introduction. As mentioned, such algorithms can perform well also in a more difficult semi-adversarial setting. Before we present these algorithms let us first describe formally the semi-adversarial variant.

Definition 2.14: The Semi-Adversarial Model

As in the purely-probabilistic Noisy Advice Model, each node is chosen to be *faulty* with probability q, and otherwise it is *sound*. Also, similarly to the original model, a sound vertex always points at its correct neighbors. However, in the semi-adversarial model, a faulty node u no longer points at a neighbor chosen uniformly at random, and instead, the neighbor w which such a node points at is chosen by an adversary. Importantly, for each node u, the adversary must specify its potentially faulty advice w, before it is known which nodes will be faulty. In other words, first, the adversary specifies the faulty advice w for each node u, and then the environment samples which node is faulty and which is sound.

2.4.1 Lower Bound in the Semi-Adversarial Variant

The following result implies that if $q > 1/\Delta$ then any algorithm must have exponential query and move complexity in the depth D.

Theorem 2.15

Consider an algorithm in the semi-adversarial model. On the complete Δ-ary tree of depth D, the expected number of queries to find the treasure is $\Omega\left((q\Delta)^D\right)$. The lower bound holds even if the algorithm has access to the advice of all internal nodes in the tree.

We first need a simple observation, that follows from Yao's principle (see [31] for a proof).

Observation 2.16

Any randomized algorithm trying to find a treasure chosen uniformly at random between k identical objects will need an expected number of queries that is at least $(k+1)/2$.

Proof of Theorem 2.15. Consider the complete Δ-ary tree and assume that the treasure is located at a leaf. The adversary behaves as follows. For any advice it gets a chance to manipulate, it would always make it point towards the root. With probability q^D the adversary gets to choose all the advice on the path between the root and the treasure. Any other advice points towards the root as well (either because it was correct to begin with or because it was set by the adversary). Hence with probability q^D the tree that the algorithm sees is the same regardless of the position of the treasure. It follows from Observation 2.16 that the time to find the treasure can only be linear in the number of leaves which is $\Omega(\Delta^D)$. □

2.4.2 Probabilistic Following Algorithms

Recall that a *Probabilistic Following* (PF) algorithm is specified by a *listening* parameter $\lambda \in (0,1)$. At each step, the algorithm "listens" to the advice with probability λ and takes a uniform random step otherwise. The first item in the next theorem states that if the noise parameter is smaller than c/Δ for some small enough constant $0 < c < 1$, then there exists a listening parameter λ for which Algorithm PF achieves $\mathcal{O}(d)$ move complexity. Moreover, this result holds also in the semi-adversarial model. Hence, together with Theorem 2.15, it implies that in order to achieve efficient search, the noise parameter threshold for the semi-adversarial model is $\Theta(1/\Delta)$.

Theorem 2.17

1. Assuming $q_u < 1/(10\Delta_u)$ for every u, then PF with parameter $\lambda \in [0.7, 0.8]$ finds the treasure in less than $100d$ expected steps, even in the semi-adversarial setting.
2. Consider the complete Δ-ary tree and assume that $q > 10/\Delta$. Then for any choice of λ the hitting time of the treasure by PF is exponential in the depth of the tree, even assuming the faulty advice is drawn at random.

Proof. Our plan is to show that the expected time to make one step in the correct direction is $\mathcal{O}(1)$, from any starting node. Conditioning on the advice setting, we make use of the Markov property to relate these elementary steps to the total travel time. The main delicate point in the proof stems from dealing with two different sources of randomness. Namely the randomness of the advice and that of the walk itself.

In this section, it is convenient to picture the tree as rooted at the target node τ. For any node u in the tree, we denote by u' the parent of u with respect to the treasure. With this convention, correct advice at a node u points at u', while incorrect advice points at one of its children. The fact the walk moves on a tree means that for a given advice setting, the expected (over the walk) time it takes to reach u' from u can be written conveniently as a product of a variable involving the advice at u only and the advice on the set of u's descendants (the two being independent).

We denote by $t(u)$ the time it takes to reach node u. Manipulating average symbols such as $\mathbb{E}$ requires extra care. Indeed, there are two sources of randomness, the first being the randomness used in drawing the advice and the second being the randomness used in the walk itself. We write $\mathbb{E}$ for averaging over the advice, while we use E_u to denote expectation over the walk, conditioning on u being the starting node. As a remark, observe that $E_u(t(v))$ depends on the advice configuration, it is a random variable with respect to the advice, while $\mathbb{E}E_u(t(v))$ really is just a number.

The following is the central lemma of this section.

Lemma 2.18

Assume that for every vertex u, $q_u < 1/(10\Delta_u)$, and $\lambda \in [0.7, 0.8]$. Then for all nodes u, $\mathbb{E}E_u t(u') \leq 100$. The result holds also in the semi-adversarial model.

Let us now see how we can conclude the proof of the first item in Theorem 2.17, given the lemma. Consider a designated source σ. Let us denote by $\sigma = u_d, u_{d-1}, \ldots, u_0 = \tau$ the nodes on the path from σ to τ. Let δ_i be the random variable indicating the time it takes to reach u_{i-1} after u_i has been visited for the first time. With these notations, the time to reach τ from σ is precisely $\sum_{i=1}^{d(\sigma,\tau)} \delta_i$. Hence, the expected time to reach τ from σ is $\sum_{i=1}^{d(\sigma,\tau)} \mathbb{E}[E_\sigma \delta_i]$.

Conditioning on the advice setting, the process is a Markov chain and we may write

$$E_\sigma \delta_i = E_{u_i} t(u_{i-1}).$$

Taking expectations over the advice ($\mathbb{E}$), under the assumptions of Lemma 2.18, it follows that $\mathbb{E}(E_\sigma \delta_i) \leq 100$, for every $i \in [d(\sigma, \tau)]$. And this immediately implies a bound of $100 \cdot d(\sigma, \tau)$.

Proof of Lemma 2.18. We start with partitioning the nodes of the tree according to their distance from the root τ. More precisely, for $i = 1, 2, \ldots, D$, where D is the depth of the tree, let us define

$$\mathcal{L}_i := \{u \in T : d(u, \tau) = i\} .$$

The nodes in $\mathcal{L}_i$ are referred to as *level-i* nodes. We treat the statement of the lemma for nodes $u \in \mathcal{L}_i$ as an induction hypothesis, with i being the induction parameter. The induction goes backwards, meaning we assume the assumption holds at level $i + 1$ and show it holds at level i. The case of the maximal level (base case for the induction) is easy since, at a leaf the walk can only go up and so if u is a leaf $\mathbb{E}E_u(t(u')) = 1 < 100$.

Assume now that $u \in \mathcal{L}_i$. We first condition on the advice setting. A priori, $E_u\tau(u')$ depends on the advice over the full tree, but in fact it is easy to see that only advice at layers $\geq i$ matter. Recall from Markov Chain theory that an *excursion* to/from a point is simply the part of the walk between two visits to the given point. We denote L_u the average (over the walk only) length of an excursion from u to itself that does not go straight to u' and we write N_u to denote the expected (over the walk only) number of excursions before going to u'. We also refer to this number as a number of *attempts*. The variable N_u can be 0 if the walk goes directly to u' without any excursion. We decompose $t(u')$ in the following standard way, using the Markov property

$$E_u t(u') = 1 + L_u \cdot N_u. \tag{2.1}$$

Indeed the expectation $E_u t(u')$ can be seen as the expectation (over the walk) of $1 + \sum_{i=1}^{T} Y_i$ where the Y_i's are the lengths of each excursion from u and T is the (random) number of such excursions before hitting u'. The term 1+ accounts for the step from u to u'. The event $\{T \geq t\}$ is independent of $Y_1, \ldots, Y_t$ and so using Wald's identity we have that $E_u t(u') = 1 + E_u T \cdot E_u Y_1$. The term $E_u T$ is equal to N_u (by definition) while $E_u Y_1$ is equal to L_u (by definition).

We now want to average equality (2.1), which is only an average over the walk, by taking the expectation over all advice in layers $\geq i$. To this aim, we write L_u as follows

$$L_u = 1 + \sum_{v \neq u', v \sim u} p_{u,v} E_v t(u),$$

where we write $u \sim v$ when u and v are neighbors in the tree and $p_{u,v}$ is the probability to go straight from u to v given the advice setting. By assumption on the model, $E_v t(u)$ depends on the advice at layers $\geq i + 1$ only, if we start at a node $v \in \mathcal{L}_{i+1}$, while both $p_{u,v}$ and N_u

depend only on the advice at layer $= i$ of the tree. This is true also in the semi-adversarial model. Hence when we average, we can first average over layers $> i$ to obtain, denoting $\mathbb{E}^{>i}$, the expectation over the layers $> i$,

$$\begin{aligned}\mathbb{E}^{>i}E_u t(u') &= 1 + \left(1 + \sum_{v\neq u', v\sim u} p_{u,v}\mathbb{E}^{>i}E_v t(u)\right) N_u, \\ &= 1 + \left(1 + \sum_{v\neq u', v\sim u} p_{u,v}\mathbb{E}E_v t(u)\right) N_u. \end{aligned} \tag{2.2}$$

and using the fact that,$\sum_{v\neq u'} p_{u,v} \leq 1$, together with the induction assumption at rank $i+1$, we obtain

$$\mathbb{E}^{>i}E_u t(u') \leq 1 + (1+100)\, N_u.$$

From now on we replace 100 by a parameter $\kappa > 0$, for mere aesthetic reasons. Averaging over the layer i of advice we obtain

$$\mathbb{E}E_u t(u') \leq 1 + (1+\kappa)\, \mathbb{E}N_u.$$

It only remains to analyse the term $\mathbb{E}N_u$. If the advice at u is correct, which happens with probability $p_u = 1 - q_u$, then the number of attempts follows a (shifted by 1) geometric law with parameter $\lambda + \frac{(1-\lambda)}{\Delta_u}$. In words, when the advice points to u' which happens with probability at most 1, the walker can go to the correct node either because she listens to the advice, which happens with probability λ, or because she did not listen, but still took the right edge, which happens with probability $\frac{(1-\lambda)}{\Delta_u}$. Similarly, when the advice points to a node $\neq u'$, which happens with probability at most q_u, then N_u follows a geometric law (shifted by 1) with parameter $\frac{(1-\lambda)}{\Delta_u}$. The conclusion is that

$$\begin{aligned}\mathbb{E}N_u &\leq \left(\frac{1}{\lambda + \frac{(1-\lambda)}{\Delta_u}} - 1\right) + q_u\left(\frac{\Delta_u}{1-\lambda} - 1\right) \\ &\leq \frac{1}{\lambda} - 1 + \frac{q_u\Delta_u}{1-\lambda} \end{aligned} \tag{2.3}$$

And so it follows that

$$\mathbb{E}E_u t(u') \leq 1 + (1+\kappa)\cdot\left(\frac{1}{\lambda} - 1 + \frac{q_u\Delta_u}{1-\lambda}\right)$$

Hence if $q_u\Delta_u < 0.1$ and we choose $\lambda \in [0.7, 0.8]$ (for instance, we made no attempt in optimizing these constants), we see that $\mathbb{E}N_u < 0.8$. This is because

$$\frac{1}{\lambda} - 1 + \frac{0.1}{1-\lambda} \leq \frac{10}{7} - 1 + \frac{0.1}{1-0.8} < 0.93$$

Hence it follows that $\mathbb{E}E_u t(u') \leq 1 + 0.93(1+\kappa) < \kappa$. The last inequality holds by choice of $\kappa = 100$. By our (backwards) induction, we have just shown that, if $q < \frac{1}{10\Delta}$ and we set $\lambda \in [0.7, 0.8]$ then for all nodes u in the tree

$$\mathbb{E}E_u t(u') < 100.$$

This concludes the proof of Lemma 2.18 and hence also of the first part of Theorem 2.17. □

Let us explain how the lower bound in the second part of Theorem 2.17 is derived in the case that $q\Delta > 10$. We assume we are in a complete Δ-ary tree under our usual uniform noise model. With probability q there is fault at u and with probability $1 - \frac{1}{\Delta}$ the advice does not point to u'. In this case, N_u follows a geometric law with parameter $\frac{1-\lambda}{\Delta}$. Hence

$$\mathbb{E}(N_u) \geq q\Delta\left(1 - \frac{1}{\Delta}\right)\frac{1}{1-\lambda} - 1 \geq \frac{10(1-\frac{1}{\Delta})}{1-\lambda} - 1 \geq 10\left(1 - \frac{1}{\Delta}\right) - 1 \geq 3,$$

for any choice of λ, since $\Delta \geq 2$. We proceed very similarly, by induction, and use Equality (2.2) together with the previous bound on $\mathbb{E}(N_u)$ to obtain that for any node on layer i, u with parent u', $\mathbb{E}E_u t(u') \geq 1 + 3\min_{v\in\mathcal{L}_{i+1}} \mathbb{E}E_v t(v')$, so in particular

$$\min_{u\in\mathcal{L}_i} \mathbb{E}E_u t(u') \geq 1 + 3 \min_{v\in\mathcal{L}_{i+1}} \mathbb{E}E_v t(v').$$

The expected hitting time of the target τ, even starting at one of its children is therefore of order $\Omega(3^D)$. □

Observation 2.19

The proof uses crucially the tree structure and does not extend to general graphs straightforwardly. Specifically, on a tree there is a single path from σ to τ and so the points u_i are uniquely defined, they are not random. Moreover an excursion from a node u at Layer i that does not visit it's parent can only remain in layers $\geq i$. This was used through the fact that $E_v t(u)$ depends only on the advice at layers $\geq i$, if we start at a node $v \in \mathcal{L}_i$.

2.5 Upper Bounds in High Probability

We assume the following, w.l.o.g.

- The noise model is the semi-adversarial variant. Hence the results apply also to the random variant.
- The algorithm knows the depth d of the treasure. This assumption can be removed by an iterative process that guesses the depth to be 1,2,.... By running each iteration for a limited time as specified in the theorem, the asymptotic runtime is not violated.

Given that the algorithm knows the depth d of the treasure, we further assume w.l.o.g. that it never searches a node at depth greater than d. Equivalently, we may (and do) assume that the depth of the tree is $d = D$, and that the treasure is located at a leaf.

- The tree is balanced: all leaves are at depth D, and all non-leaf nodes have degree exactly Δ. To remove this assumption, whenever the algorithm visits a node v at depth $i < D$ of degree $d_v < \Delta$, it can connect to it $\Delta - d_v$ "auxiliary trees", where each auxiliary tree has depth $D - i - 1$ and is balanced. The advice in all nodes of these auxiliary trees points towards v, which is a valid choice in the semi-adversarial model.

2.5.1 The Meta Algorithm

Underlying the upper bound presented in Theorem 2.5 is a simple, yet general, scheme. It is based on a notion of *fitness*, which we define later. This notion depends on the parameters of the model. It is carefully crafted such that the following *fitness properties* hold:

- **F1.** The fitness of a node u only depends on the advice on the path $[\sigma, u]$, excluding u.
- **F2.** For any node u on the path $[\sigma, \tau]$, $P(u \text{ is fit}) \geq 1 - \frac{\delta}{2D}$.
- **F3.** With probability at least $\geq 1 - \frac{\delta}{2}$, the connected component of fit nodes that contains the root is of size bounded by $f(D, \delta)$, for some function f.

Once fitness is appropriately defined so that properties F1 - F3 hold, an algorithm is naturally associated to it. It consists in exploring in a depth-first fashion the connected component of fit nodes containing the root. Property F1 ensures that this strategy is well-defined. The time to explore a component is at most twice its size, because each edge is traversed at most twice.

Claim 2.20

Property F2 implies that A_{walk} eventually finds τ with probability $\geq 1 - \frac{\delta}{2}$.

Proof. Using Property F2, the probability that all nodes on the root to treasure path $[\sigma, \tau]$ are fit is at least as large as $1 - \frac{\delta \cdot D}{2D} = 1 - \frac{\delta}{2}$. Under this event, τ belongs to the same component of fit nodes as the root, and hence A_{walk} eventually finds it. □

By Property F3, the A_{walk} algorithm needs a number of steps which is upper bounded by $2f(D, \delta)$ with probability $1 - \frac{\delta}{2}$. Using a union bound we derive the following claim.

Claim 2.21

If the fitness properties F1-F3 are satisfied, then A_{walk} finds τ in at most $2f(D, \delta)$ steps with probability $\geq 1 - \delta$.

2.5.2 Upper Bound in the Walk Model with High Probability

This subsection is devoted to the proof of Theorem 2.5. Our algorithm A_{walk} follows the general scheme presented in Section 2.5.1. It is based on a notion of fitness presented below. With this notion in hand, A_{walk} simply visits, in a depth-first fashion, the component of fit nodes to which the root σ belongs to.

Definition 2.22: Advice-fitness

Let $h_1 = \frac{2}{\varepsilon}\log_\Delta(4\delta^{-1}D)$ and $h_2 = \frac{6}{\varepsilon^2}\log_\Delta(4\delta^{-1}D)$. Let u be a node and u_{-h_2} be the ancestor of u at distance h_2 from u, or σ if u is at distance $< h_2$ from σ. The node u is said to be *fit* if the number of locations on the path $[u_{-h_2}, u]$ that do not point towards u is less than h_1. It is said to be *unfit* otherwise. Moreover, a fit node is said to be *reachable* if it is in the connected component of fit nodes that contains the root (as in Property F3). Equivalently, a node is reachable if either it is the root, or it is fit and its parent is reachable.

Note that by definition, all nodes at depth $< h_1$ are fit and reachable. The notion of fitness clearly satisfies the first fitness property F1. We want to show that it also satisfies Properties F2 and F3 with $f(D,\delta) = (\delta^{-1}D)^{O(1/\varepsilon)}$. Let us first give two useful conditions satisfied by our choice of h_1 and h_2.

Claim 2.23

The following inequalities hold $(*)\ 2^{h_2}\Delta^{-\varepsilon h_1} \le \frac{\delta}{4D}$, $(**)\ 2^{h_2}\Delta^{(1+\varepsilon)h_1-\varepsilon h_2} \le \frac{\delta}{4D}$.

Proof. Equation $(*)$. Replacing $h_{1/2}$ by their values, we bound the left hand side in Equation $(*)$ as follows $2^{\frac{6}{\varepsilon^2}\log_\Delta(4\delta^{-1}D)}\Delta^{-\varepsilon\frac{2}{\varepsilon}\log_\Delta(4\delta^{-1}D)} \le 2^{(6\varepsilon^{-2}-2\log\Delta)\log_\Delta(4\delta^{-1}D)}$. We assume that $\Delta \ge 2^{6\varepsilon^{-2}}$ so that $6\varepsilon^{-2} \le \log\Delta$ and $6\varepsilon^{-2} - 2\log\Delta \le -\log\Delta$. Hence the left hand side in $(*)$ is no greater than $2^{-\log\Delta\log_\Delta(\delta^{-1}D)} \le \frac{\delta}{4D}$.

Equation $(**)$. Using the fact that $\varepsilon \le 1$ and that $h_2 = \frac{3}{\varepsilon}h_1$, we obtain that $(1+\varepsilon)h_1 - \varepsilon h_2 \le 2h_1 - 3h_1 = -h_1 \le -\varepsilon h_1$. The result follows from Equation $(*)$. □

Lemma 2.24

The notion of advice-fitness obeys Property F3 $f(D,\delta) = (\delta^{-1}D)^{O(\varepsilon^{-1})}$. Hence the move complexity of A_{walk} is less than $(\delta^{-1}D)^{O(\varepsilon^{-1})}$, with probability $\ge 1 - \frac{\delta}{2}$.

Proof. Let Fit be the connected set of reachable nodes (as defined in Definition 2.22). Our goal is to show that with high probability, namely, with probability at least $1 - \frac{\delta}{2}$, we have $|\text{Fit}| = (\delta^{-1}D)^{O(\varepsilon^{-1})}$.

For $i \ge 0$, the term *i-node* will refer to any node whose common ancestor with τ is at

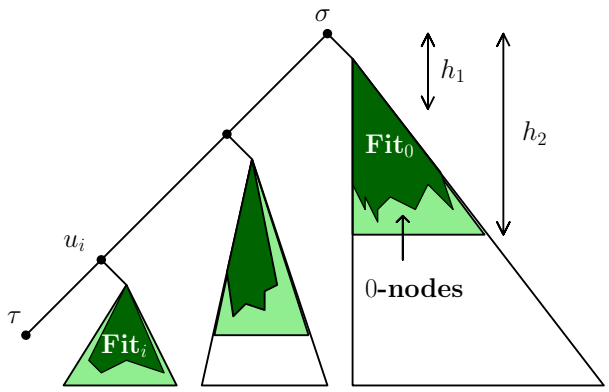

Figure 2.2: The partition of fit vertices introduced in the proof of Lemma 2.24. The colored nodes in the subtree on the right are the close 0-nodes, where those colored with dark green are the reachable fit 0-nodes. There are no fit 0-nodes at depth h_2 in this example.

depth i. An i-node is further said to be *close* if its depth d lies in the range $[i, i + h_2]$. Let Fit_i be the set of close i-nodes in Fit (see Figure 2.2).

Our first goal is to show that with high probability, Fit does not contain any 0-node at depth h_2 (Claim 2.26). Under this high probability event, A_{walk} visits only fit 0-nodes that are close (i.e., at depth at most h_2), because A_{walk} visits only reachable nodes, and fit 0-nodes that are not close are disconnected from the root at depth h_2. Hence all the 0-nodes visited by A_{walk} are in Fit_0. By symmetry, a similar statement holds for each layer i, and the corresponding subset Fit_i. Thus, under a high probability event, the nodes visited by A_{walk} during its execution form a subset of $\bigcup_{i=0}^{D} \mathrm{Fit}_i$ (namely, $f(D, \delta) \leq |\bigcup_{i=0}^{D} \mathrm{Fit}_i|$).

Denote the expected number of fit 0-nodes at depth h by

$$\alpha_h := \sum_{u \text{ is a 0-node at depth } h} P(u \text{ is fit}).$$

We have

$$\mathbb{E}\left(|\mathrm{Fit}_0|\right) \leq \sum_{u \text{ is a close 0-node}} P(u \text{ is fit}) = \sum_{h \leq h_2} \alpha_h. \tag{2.4}$$

Claim 2.25

(∗) $\sum_{h \leq h_1} \alpha_h \leq 2\Delta^{h_1}$. (∗∗) For any $h_1 < h \leq h_2$, we have $\alpha_h \leq \Delta^{h_1(1+\varepsilon)} 2^h \Delta^{-\varepsilon h}$.

Proof. Every node at depth at most h_1 is fit. There are at most $2\Delta^{h_1}$ such nodes. Estimation (∗) follows.

We now show the second estimate. Let U_h be a node chosen uniformly at random among all 0-nodes at depth h. Then $P(U_h \text{ is fit}) = \sum_{u \text{ is a close 0-node}} P(U_h = u)P(u \text{ is fit})$, and hence we may write:

$$\alpha_h = (\Delta - 1)^h P(U_h \text{ is fit}).$$

Choosing U_h randomly rather than arbitrarily is of crucial importance in the semi-adversarial variant, because the adversary might choose to direct all the faulty advice towards a specific node u. This could result in some terms in the sum (in the definition of α_h) being much bigger than the average. So it is important to avoid bounding the average by the max. We draw a uniform path $\sigma = U_0, U_1, \ldots, U_h$ of length h from the root, in the component of 0-nodes. Consider a node U_i on this path. With probability q, it is faulty. In this case, regardless of how the adversary could set its advice, there is only probability of at most $\frac{1}{\Delta-1}$ over the choice of U_{i+1} that the advice at U_i points to U_{i+1}.

It follows that the number of ancestors of U_h whose advice points to U_h may be viewed as the sum of h Bernoulli variables B_i with parameter $\frac{q}{\Delta-1}$. Moreover, the previous argument means that $P(B_i = 1 \mid B_j, j < i) = \frac{q}{\Delta-1} = P(B_i = 1)$. The B_i variables are thus uncorrelated and hence independent since they are Bernoullis. The node U_h is fit if at least $h - h_1$ ancestors point to it. Thus, by a union bound over the $\binom{h}{h-h_1} \leq 2^h$ possible locations of faults, $P(U_h \text{ is fit}) \leq 2^h \left(\frac{q}{\Delta-1}\right)^{h-h_1}$. Hence

$$\alpha_h \leq 2^h(\Delta - 1)^h \left(\frac{q}{\Delta - 1}\right)^{h-h_1} \leq 2^h q^{h-h_1} \Delta^{h_1} = \Delta^{h_1(1+\varepsilon)} 2^h \Delta^{-\varepsilon h}.$$

In the last step, we used $q = \Delta^{-\varepsilon}$. This concludes the proof of Claim 2.25. □

For $h = h_2$, the combination Claim 2.25 and Equation $(**)$ stated in Claim 2.23 implies:

$$\alpha_{h_2} \leq \frac{\delta}{4D} \tag{2.5}$$

For $i \in [D]$, denote by Z_i (Z for *zero*) the event that there are no fit i-nodes at depth $i + h_2$. Applying Markov inequality on Equation (2.5) implies that $P(Z_0) \geq 1 - \delta(4D)^{-1}$. Since the same argument can be applied to any $i \leq D$, we get

Claim 2.26

For any $i \leq D$, we have $P(Z_i) \geq 1 - \frac{\delta}{4D}$ and hence $P(\bigcap Z_i) \geq 1 - \frac{\delta}{4}$.

If follows from the assumption on $\Delta \geq 2^{6\varepsilon^{-2}}$ that $\Delta^\varepsilon \geq 2^6 \geq 8$, so that $2\Delta^{-\varepsilon} < \frac{1}{4}$. Using

Claim 2.25 and the definition of h_1 we get:

$$\begin{aligned}\mathbb{E}\left(|\mathrm{Fit}_0|\right) &\leq \sum_{h\leq h_2} \alpha_h \leq 2\Delta^{h_1} + \sum_{h=h_1}^{h_2} \Delta^{h_1(1+\varepsilon)} \left(2\Delta^{-\varepsilon}\right)^h \\ &\leq 2\Delta^{h_1} + \Delta^{h_1(1+\varepsilon)} \sum_{h\geq h_1} 4^{-h} \leq 2\Delta^{h_1} + 2\Delta^{h_1(1+\varepsilon)} \\ &\leq 4\cdot(4\delta^{-1}D)^{2\varepsilon^{-1}+2} = (4\delta^{-1}D)^{2\varepsilon^{-1}+3}.\end{aligned}$$

The computation is the same for any $i \leq D$, yielding that $\mathbb{E}(|\mathrm{Fit}_i|) \leq (4\delta^{-1}D)^{2\varepsilon^{-1}+3}$, and by linearity of expectation, we obtain $\mathbb{E}(|\bigcup_{i=0}^{D} \mathrm{Fit}_i|) \leq D\cdot(4\delta^{-1}D)^{2\varepsilon^{-1}+3}$. Using the Markov inequality, with probability at least $1-\frac{\delta}{4D}$, this variable is upper bounded by $4\delta^{-1}D^2\cdot(4\delta^{-1}D)^{2\varepsilon^{-1}+3} \leq (4\delta^{-1}D)^{2\varepsilon^{-1}+5}$. As explained in the beginning of the proof, under the event $\bigcap_{i\leq D} Z_i$ (which occurs with probability at least $1-\frac{\delta}{4}$ thanks to Claim 2.26), we have $\mathrm{Fit} \subset \bigcup_{i\leq D} \mathrm{Fit}_i$. Using a union bound, we conclude that with probability at least $1-\frac{\delta}{2}$, we have $|\mathrm{Fit}| \leq (4\delta^{-1}D)^{2\varepsilon^{-1}+5} = (\delta^{-1}D)^{O(\varepsilon^{-1})}$, as desired. □

Claim 2.21 in combination with Lemma 2.24 proves Theorem 2.5.

2.6 Lower Bound

The goal of this section is to prove Theorem 2.6. The proof is done for the query complexity, in a complete Δ-ary tree of depth $D = \log_\Delta n$. The proof for the move complexity immediately follows. Throughout the proof, T denotes a complete Δ-tree of depth D. In our lower bound, the adversary places τ at a leaf of T chosen at random from the uniform distribution. We denote by F the set of faulty locations (without directional advice). Since this set as well as τ are chosen uniformly at random, we may assume without loss of generality that the algorithm is deterministic. The presentation of our proof is simplified if we assume that the algorithm is told which nodes of T are faulty (namely, are in F). We can make this assumption because it only strengthens our lower bound.

The letter H is reserved to denote a subtree of T. We consider implicitly that a subtree H only contains the descendants of its root (with respect to the original root σ). A subset S is said to be *completely faulty* if all nodes in S are faulty. Overloading this expression, we say that a leaf v of some subtree H is *completely faulty* if the path from the root of H towards v is completely faulty. The relevant reference subtree H will be specified if it is not clear from context. The number of completely faulty leaves of a subtree H is denoted $B(H)$ or simply B if H is clear from context. When considering a subset $S \subseteq T$, we write S^* to denote the pair $(S, S\cap F)$. In words, this corresponds to a subset with the information of which nodes are faulty.

2.6.1 Proof of Theorem 2.6

At a high level, the argument is as follows. On the path from the root to τ, typically, there exists a segment $[v_i, v_{i+h-1}]$ of length approximately $h := \frac{1}{\varepsilon}\log_\Delta D$, where all advice is faulty (Lemma 2.27), . On the other hand, the tree rooted at v_i of depth h, typically hosts many such completely faulty leaves (Lemma 2.28). In some sense these leaves are *indistinguishable*. This means that any algorithm, will need to try a constant fraction of them before finding v_{i+h}, the one leading to τ with constant probability.

Let us make this intuition more precise. For each choice of faulty locations F and treasure location $\tau = v$, we define $u(v, F)$ to be the first node on the path $[\sigma, v]$ such that u and its $h-1$ descendants towards v (the next $h-1$ nodes on the path to v) are faulty, if such u exists, and otherwise we say $u(v, F)$ is *not defined*. Denote by $\mathbf{H}(v, F)$ the subtree rooted at u of depth h. A central object in the proof is $\mathbf{H}^*(v, F)$ which corresponds to the couple $\mathbf{H}(v, F), (F \cap \mathbf{H}(v, F))$ (the subtree together with the faulty locations on it). Henceforth, we will often drop the dependency on v, F in the interest of readability, but we keep the bold notation to emphasize that $\mathbf{H}$ is a random object (it depends on F). If u is not defined, we also say $\mathbf{H}$ is not defined.

The following lemma lower bounds the probability that $\mathbf{H}$ is well defined.

Lemma 2.27

If h satisfies $q^h \geq \frac{8\delta h}{D}$ and $D \geq \max[h, 8\delta h]$ (for D that does not satisfy this condition the statement does not make sense), then $P(\mathbf{H} \text{ is well defined}) \geq 4\delta$.

Proof. Recall that we write $[\sigma, \tau] := \{v_0 = \sigma, v_1, \ldots, v_{D-1}, v_D = \tau\}$. For a given $i, h \in \mathcal{N}$, let us denote by $F_{i,h}$ the event that $[v_i, v_{i+h-1}]$ is completely faulty.

With this definition, $\mathbf{H}$ is well defined if the event $F_{i,h}$ holds for at least one value of i in the range $0 \leq i \leq D-h$. Hence what we want to show is that

$$P\left(\text{NOT} \bigcup_{i=0}^{D-h} F_{i,h}\right) \leq 1 - 4\delta. \tag{2.6}$$

For any fixed, i, h, $P(F_{i,h}) = q^h$. Indeed, there are h nodes on the path $[v_i, v_{i+h-1}]$ and each is independently faulty with probability q. The parameter h satisfies $q^h = \frac{8\delta h}{D}$ by assumption.

The events $F_{i'h,h}$ are independent when i' varies in $[D/h]$. The probability that none of these holds is

$$(1-q^h)^{D/h} \leq (1 - \frac{8\delta h}{D})^{\frac{D}{h}} \leq e^{-\frac{8\delta h}{D}\frac{D}{h}} = e^{-8\delta} \leq 1 - 8\delta/2 = 1 - 4\delta.$$

The last inequality holds for sufficiently small δ (e.g., $\delta \leq \frac{1}{16}$). □

From now on, we assume for simplicity that h is chosen so that $q^h = \frac{8\delta h}{D}$. ($h$ being an integer, this is only an approximate equality in general. We ignore this point in the discussion,

assuming h has been appropriately rounded.) Recall that $q = \Delta^{-\varepsilon}$, so taking logarithms we see that $\varepsilon h = \log_\Delta \frac{D}{\delta} - \log_\Delta 8h$. Viewing δ and Δ as fixed and letting D go to infinity, the previous equality entails that $h = \frac{1}{\varepsilon} \log_\Delta \frac{D}{\delta}(1 - o(\cdot))$, where the term $o(\cdot)$ is a function of D going to 0 when D tends to infinity[4].

Let $B(\mathbf{H}^*)$ denote the number of completely faulty leaves in $\mathbf{H}^*$. Lemma 2.28 below is proven in Section 2.6.2. It bounds the typical value of $B(\mathbf{H}^*)$ in those cases that $\mathbf{H}^*$ is well defined.

Lemma 2.28

Let C be a sufficiently large constant. Then

$$P\left(B(\mathbf{H}^*) \leq (q\Delta)^{h-C} \mid \mathbf{H} \text{ is well defined}\right) \leq 0.5.$$

The following two intermediate results express how "indistinguishable" is formalized in this context.

Claim 2.29

Consider a leaf v and a subtree $\mathbf{H}^*$ (together with the faulty locations in it). Then, the value of $p_{v,H^*} := P_F(\mathbf{H}^*(v, F) = H^* \mid \tau = v)$ is the same for all v such that $p_{v,H^*} > 0$.

Proof. Let H be a fixed subtree of depth h rooted at some node u. By definition, the statement that $P(\mathbf{H}^*(v, F) = H^* \mid \tau = v) > 0$ is equivalent to the following two statements:

$\mathcal{A}$: On the path $[\sigma, u]$, there are no h consecutive faulty nodes and, if $u \neq \sigma$, the parent of u is not faulty.

$\mathcal{B}$: The leaf v is a descendant of u and the leaf of H^* which is an ancestor of v is completely faulty in H^*.

The probability of $\mathcal{A}$ depends only (in some complicated way) on the length of $[\sigma, u]$ and q and hence does not depend on v. With these notations, if v,H^* and F are such that $\mathcal{A}$ and $\mathcal{B}$ hold, then

$$P(\mathbf{H}^*(v, F) = H^* \mid \tau = v) = q^{|F \cap H|}(1 - q)^{|H \setminus F|} P(\mathcal{A}).$$

The right hand side does not depend on v. The claim follows. □

Lemma 2.30

Conditioning on the subtree $\mathbf{H}^*$ (and hence its existence), the leaf of $\mathbf{H}^*$ which leads to the treasure is uniform amongst all completely faulty leaves v of $\mathbf{H}^*$.

[4] Indeed, h goes to infinity when D goes to infinity so $\log_\Delta 4h = o(h)$.

Proof. Denote by $\mathcal{L}$ the set of leaves of T. Using Bayes rule, and because we assume that τ is uniform over all leaves $\mathcal{L}$,

$$P(\tau = v \mid \mathbf{H}^*) = P(\tau = v) \cdot \frac{P(\mathbf{H}^* \mid \tau = v)}{P(\mathbf{H}^*)} = \frac{1}{|\mathcal{L}|} \cdot \frac{P(\mathbf{H}^* \mid \tau = v)}{P(\mathbf{H}^*)}.$$

It follows that $P(\tau = v \mid \mathbf{H}^*)$ has the same value for all leaves v of T such that $P(\tau = w \mid \mathbf{H}^*) > 0$. Indeed, we saw that the right hand term is independent of w, as soon as w descends from a completely faulty leaf of $\mathbf{H}^*$ (Claim 2.29), and otherwise it is 0.

Since the tree T is complete and regular, each leaf of $\mathbf{H}^*$ is the ancestor of the same number of leaves in T, and so each completely faulty leaf of $\mathbf{H}^*$ is equally likely to lead to the treasure. □

We now condition on the event that $\mathbf{H}$ is well defined and that it has more than $s = (q\Delta)^{h-C}$ completely faulty leaves. This event holds with probability at least $4\delta \times 0.5$ (combining the results of Lemma 2.27 and Lemma 2.28). Under this conditioning, with probability at least 0.5 over treasure location the completely faulty leaf leading to the treasure is visited after at least $0.5s$ other faulty leaves have been visited. Indeed there are s faulty leaves, each being equally likely to lead to the treasure (Lemma 2.30). Overall, with probability $4\delta \times 0.5 \times 0.5 = \delta$, more than $0.5s$ nodes need to be visited. We saw that, $h = \frac{1}{\varepsilon}\log_\Delta(\frac{D}{\delta})(1-o(*))$, hence $s = (q\Delta)^{h-C} = (q\Delta)^{h\cdot(1-o(*))} = (\Delta^{1-\varepsilon})^{\frac{1}{\varepsilon}\log_\Delta \frac{D}{\delta}(1-o(*))}$. After simplification, this is $(\delta^{-1}D)^{\frac{1-\varepsilon}{\varepsilon}(1-o(*))}$.

2.6.2 Proof of Lemma 2.28

The proof of Lemma 2.28 is broken into intermediate claims. To begin with we ignore the treasure τ, and consider a fixed complete Δ-ary tree H of depth h. Each node of H is faulty (namely, belongs to the set F) independently with probability q.

Claim 2.31

It holds that $P\left(B(H) > \frac{1}{2}(q\Delta)^h\right) = \Omega(q)$.

Proof. The proof uses a second moment argument. For any given leaf, the probability of the full path from the root being faulty is q^h and there are Δ^h leaves. Hence $\mathbb{E}(B) = (q\Delta)^h$. Let us denote by $\mathcal{L}$ the set of leaves. We may estimate $\mathbb{E}(B^2)$ using

$$\mathbb{E}(B^2) = \mathbb{E}(B) + \sum_{u \neq v \in \mathcal{L}} P(u \text{ and } v \text{ are completely faulty}).$$

Fix a leaf $v \in \mathcal{L}$. We partition other leaves according to the depth $h-\ell$ of their common ancestor with v. Let us denote such leaves $\mathcal{L}_\ell$. For any $\ell \in [1, h]$, $|\mathcal{L}_\ell| \leq \Delta^\ell$. Moreover, for $u \in \mathcal{L}_\ell$, u and v are completely faulty is equivalent to v being completely faulty and the $\ell - 1$ nodes connecting

u to the root to v path being faulty. Hence, $P(u \text{ and } v \text{ are completely faulty}) = q^{\ell+h-1}$. Altogether,

$$\mathbb{E}(B^2) \leq (q\Delta)^h + \Delta^h \sum_{\ell=1}^{h} \Delta^\ell q^{\ell+h-1} = O\left(q^{-1}(q\Delta)^h \sum_{\ell=1}^{h} (q\Delta)^\ell\right)$$
$$= O\left(q^{-1}(q\Delta)^{2h}\right) = O(q^{-1}\mathbb{E}(B)^2).$$

Using the Paley-Zygmund inequality, we get $P(B \geq \frac{1}{2}\mathbb{E}(B)) \geq \frac{1}{4}\frac{\mathbb{E}(B)^2}{\mathbb{E}(B^2)} = \Omega(q)$. □

Claim 2.32

For a constant C that depends only on q, for h that goes to infinity with D, and for sufficiently large D, it holds that $P(B \leq (q\Delta)^{h-u(h)} \mid B \geq 1) \leq 0.5$.

Proof. Since $B \geq 1$ there exists a path $[\sigma, v]$ which is completely faulty. For every $u \in [\sigma, v]$ define T_u as the subtree rooted at u excluding the subtree rooted at the child of u on $[\sigma, v]$. The subtrees T_u are pairwise disjoint and form a partition of T. For $u \in [\sigma, v]$, define $B_u := B(T_u)$, the number of completely faulty leaves on T_u. Note that $B = \sum_u B_u \geq \max_u B_u$. Moreover since, for $u \neq u'$, $T_u \cap T_{u'} = \emptyset$, the variables (B_u) are independent.

Let S be the prefix of size $u(h)$ of $[\sigma, v]$. All subtrees rooted at a node $u \in S$ are of depth $> h - u(h)$. Using Claim 2.31, together with the independence of the B_u's, the probability that all B_u's are smaller than $(q\Delta)^{h-u(h)}$ is less than $(1-cq)^{u(h)}$ for a small constant c. The result follows, if C is big enough (as a function of q). □

If it exists, by definition, $\mathbf{H}^*$ has at least one completely faulty leaf, which is the one leading to τ. Outside of the branch leading to τ, the nodes of $\mathbf{H}^*$ are still independently faulty with probability q. This means that the number of completely faulty leaves of $\mathbf{H}^*$, $B(\mathbf{H}^*) \mid \{\mathbf{H}^* \text{ is well defined}\}$ is distributed as $B(H^*) \mid \{B(H^*) \geq 1\}$ for any fixed subtree H of depth h.

Using this together with Claim 2.32 finishes the proof of Lemma 2.28. Indeed, we obtain

$$P\left(B(\mathbf{H}^*) \leq (q\Delta)^{h-u(h)} \mid \mathbf{H} \text{ is well defined}\right) \leq 0.5.$$

2.7 Open Problems

Obtaining efficient search algorithms for general graphs is highly intriguing. Even though the likelihood of a node being the treasure under a uniform prior can still be computed in principle, it is not so easy to compare two nodes as in Theorem 2.9 because there may be more than a single path between them.

In a limited regime of noise, we believe that memoryless strategies might very well be efficient also on general graphs, and we pose the following conjecture. Proving it may require

the use of tools from the theory of RWRE, which seem to be lacking in the context of general graph topologies.

Conjecture 2.7.1. *There exists a probabilistic following algorithm that finds the treasure in expected linear time on any undirected graph assuming $q < c/\Delta$ for a small enough $c > 0$.*

Chapter 3

Searching a Tree with Noisy Local Advice: Query Complexity

3.1 Introduction

In the previous chapter, we introduced a model for tree search with Local Noisy Advice. The goal is to reach a target τ using local guiding instructions at each node of the tree, which can be faulty with some fixed probability $q \in [0, 1]$. We studied algorithms that worked well in expectations and others that were only guaranteed to be efficient with some probability.

The algorithmic cost was measured through the number of edge traversals. In this chapter we consider another complexity cost, the query complexity. It corresponds to the number of advice that needs to be revealed. An important difference with respect to the previous chapter is we now assume the tree is known in advance to the algorithm, so it may pre-compute a list of nodes to query. These are typically *separators*, i.e., nodes that separate the tree into roughly equal size subtrees and thus allow to make great progress in the search.

In the previous chapter, query complexity was already used, but in lower bounds only. If, there were no mistakes in the advice ($q = 0$) a separator (as formally defined below) based strategy would yield $O(\log n)$ queries worst-case strategies. Our goal is to achieve similar complexities, even in the presence of faults.

3.2 Notation

The notation is the same as in the previous chapter. The target is denoted τ, while σ stands for the root of the tree (previously, the starting point of the search, here it is an just an arbitrary vertex). We use the letter n for the size of the tree. Moreover, in this chapter, we use the notation $\mathcal{Q}(\mathtt{A})$ to denote the expected query cost of algorithm $\mathtt{A}$.

	Upper Bound		Lower Bound	
	Regime	Moves	Regime	Moves
Expectation	$q << \frac{1}{\sqrt{\Delta}}$	$\tilde{\mathcal{O}}(\sqrt{\Delta}\log n)$	$q >> \frac{1}{\sqrt{\Delta}}$	$n^{\Omega(1)}$
High Probability	$q = \Delta^{-\varepsilon}$	$(\log n)^{O(\varepsilon^{-1})}$	$q = \Delta^{-\varepsilon}$	$(\log n)^{\Omega(\varepsilon^{-1})}$

Figure 3.1: Query complexity results, in simplified form. The precise conditions behind the symbol $<<$ will be clarified later.

3.3 Our results

Our main result is the following. It works assuming the advice parameter is the same at every node, i.e., is bounded by the maximum degree rather than the local degree.

Theorem 3.1

Assume that $q < c/\sqrt{\Delta}$ for a small enough constant $c > 0$. Then there exists a deterministic query algorithm $\mathtt{A}_{2-\text{layers}}$ such that $\mathcal{Q}(\mathtt{A}_{2-\text{layers}}) = \mathcal{O}(\sqrt{\Delta}\log n \cdot \log\log n)$.

In fact, we will start by showing the slightly weaker bound below, as it contains the main ideas and works in the more general setting where the mistake parameter depends on the node q_u. Condition $(\star)$ was introduced in the previous chapter (see Chapter 2, Definition 2.1). It can roughly be understood as saying that for all nodes v, $q_v < \frac{1-\varepsilon}{\sqrt{\Delta_v}}$.

Theorem 3.2

For any $\varepsilon > 0$, there exists a deterministic query algorithm $\mathtt{A}_{query}$ such that if Condition $(\star)$ holds with parameter ε then the query complexity is $\mathcal{Q}(\mathtt{A}_{query}) = \mathcal{O}(\sqrt{\Delta}\log\Delta \cdot \log^2 n)$.

Combining the ideas of Theorem 3.2 with the result Theorem 2.5 presented in the previous chapter, it is possible to derive a query result also in the high probability setting.

Corollary 3.3

Under the assumptions of Theorem 2.5, assuming n is sufficiently large (as a function of ε and Δ), there exists an algorithm A_{query} in the query model that finds the treasure with probability at least $1 - \delta$ whose number of queries scales like $(\delta^{-1} \log n)^{O(\varepsilon^{-1})}$. I.e., $P[\mathcal{Q}(A_{query}) < (\delta^{-1} \log n)^{O(\varepsilon^{-1})}] > 1 - \delta$. This result holds in the semi-adversarial variant as well.

Once again, as in the previous chapter, achieving a high probability guarantee seems easier than low runtime in expectation. Anticipating on our proofs, the intuition for this is as follows. Our query strategies work provided the neighborhood of separator nodes are *well-behaved*. Given our formalization of well-behaved, this turns out a high probability event. If we are after a guarantee in expectation, it means we need to gain some control on what happens when this event does not hold, which requires extra work.

The lower bound presented in Theorem 2.3 of the previous chapter was phrased in terms of the depth D of the tree but for query algorithms already. Since the trees considered in this lower bound are complete and regular, we may replace D by $\log_\Delta n$, we obtain a $n^{\Omega(1)}$ lower bound (as given in Table 3.1) when $q > \frac{1+\varepsilon}{\sqrt{\Delta}-1}\left(1 + \frac{1}{\Delta-1}\right)$.

The following is a lower bound that holds regardless of the noise regime.

Theorem 3.4

Consider a complete Δ-ary tree and a search algorithm $\mathtt{A}$. Then $\mathcal{Q}(\mathtt{A}) = \Omega(q\Delta \log_\Delta n)$

Organization of This Chapter. We start by the proof of the lower bound in Section 3.4 The proof of Theorem 3.2 is presented in Section 3.5. We proceed to show the proof of Corollary 3.3 in Section 3.6. Section 3.7 is by far the longest section. It is entirely devoted to the proof of Theorem 3.1.

3.4 A Lower Bound of $\Omega(\sqrt{\Delta} \cdot \log_\Delta n)$ when $q \sim 1/\sqrt{\Delta}$

We now prove Theorem 3.4. Specifically, we wish to prove that for $\Delta \geq 3$, on the complete Δ-ary tree of depth D, any algorithm needs $\Omega(q\Delta D)$ queries in expectation. Note that, in particular, when q is roughly $1/\sqrt{\Delta}$, and n is the tree size, the query complexity is $\Omega(\sqrt{\Delta} \cdot \log_\Delta n)$. Before proving this lower bound, we recall Observation 2.16 from the previous chapter:

Observation

Any randomized algorithm trying to find a treasure chosen uniformly at random between k identical objects will need an expected number of queries that is at least $(k+1)/2$.

To prove the lower bound of $\Omega(q\Delta D)$, consider the complete Δ-ary tree of depth D. We prove by induction on D, that if the treasure is placed u.a.r. in one of the leaves, then the expected query complexity of any algorithm is at least $q(\Delta/2-1)D$. If $D=0$, then there is nothing to show. Assume this is true for D, and we shall prove it for $D+1$. Let $T_1,\ldots,T_{\Delta-1}$ be the subtrees hanging down from the root (in the induction, the "root" is actually an internal node, and so has $\Delta-1$ children), each having depth D. Let i be the index such that $\tau\in T_i$, and denote by Q the number of queries before the algorithm makes its first query in T_i. We will assume that the algorithm gets the advice in the root for free. Denote by Y the event that the root is faulty . In this case, Observation 2.16 applies, and we need at least $\Delta/2-1$ queries to hit the correct tree. We subtracted one query from the count because we want to count the number of queries strictly before querying inside T_i. We therefore get $\mathbb{E}[Q]\geq\mathbb{P}(Y)\cdot\mathbb{E}[Q\,|\,Y]\geq q(\Delta/2-1)$. By linearity of expectation, using the induction hypothesis, we get the result for a uniformly placed treasure over the leaves, and so it holds also in the adversarial case. □

3.5 Proof of Theorem 3.2

As is common in search on trees, our technique in this section is based on separators. We say a node u is a *separator* of T if all the connected components of $T\setminus\{u\}$ are of size at most $|T|/2$. It is well known that such a node exists. Assume there is some local procedure, that given a vertex u decides with probability $1-\delta$ in which one of the connected components of $T\setminus\{u\}$, the treasure resides. Applying this procedure on a separator of the tree, and then focusing the search recursively only on the component it pointed out, results in a type of algorithm we call a *separator based* algorithm. It uses the local procedure at most $\lceil\log_2 n\rceil$ times, and by a union bound, finds the treasure with probability at least $1-\lceil\log_2 n\rceil\delta$. Broadly speaking, we will be interested in the expected running time of this sort of algorithm conditioned on it being successful. This sort of conditioning complicates matters slightly. In what follows, we assume that the set of separators for the tree is fixed.

Proof. (of Theorem 3.1) $\mathtt{A}_{sep}$ runs a separator based algorithm in parallel (i.e. in an alternating fashion) to some arbitrary exhaustive search algorithm. Fix some small h. The local exploration procedure, denoted $\mathtt{local}_h$, for a vertex u proceeds as follows.

Procedure $\mathtt{local}_h(u)$**.** Consider the tree $T_h(u)$ rooted at u consisting of all vertices satisfying $\log_\Delta\beta(v)<h$ together with their children. So a leaf of $v\in T_h(u)$ is either a leaf of T, or satisfies $\Delta^h\leq\beta(v)<\Delta^{h+1}$. Denote the second kind a *nominee*. Call a nominee *promising* if the number of weighted arrows pointing to v is large, specifically, if $\sum_{w\in[u,v\rangle}X_w\geq\frac{2}{3}h\log\Delta$,

where $X_w = \log \Delta_w$ if the advice at w is pointing to v, $X_w = -\log \Delta_w$ if it is pointing to u, and $X_w = 0$ otherwise. Viewing it as a query algorithm, we now run the walking algorithm $\mathtt{A}_{walk}$ on $T_h(u)$ (starting at its root u) until it either finds the treasure or finds a promising nominee. In the latter case, $\mathtt{local}_h(u)$ declares that the treasure is on the connected component of $T \setminus \{u\}$ containing this nominee. If $\tau \in T_h(u)$ then set $\tau_u = \tau$. Otherwise let τ_u be the leaf of $T_h(u)$ closest to the treasure, and so in this case τ_u is a nominee. Say that u is *h-misleading* if either (1) $\tau \notin T_h(u)$ and τ_u is not promising, or (2) there is some promising nominee $v \in T_h(u)$ that is not in the same connected component of $T \setminus \{u\}$ as τ_u. In particular, if u is not h-misleading then $\mathtt{local}_h(u)$ necessarily outputs the correct component of $T \setminus \{u\}$, namely, the one containing the treasure. The proof of the following lemma is to be found below in Section 3.5.1.

Lemma 3.5

For any u, $\mathbb{P}(u \text{ is } h\text{-misleading}) \leq (\Delta + 1)(1-\varepsilon)^h$. Also, for any event X such that X occurring always implies that u is not misleading, we have $\mathbb{P}(X)\,\mathcal{Q}(\mathtt{local}_h(u) \mid X) = \mathcal{O}(\sqrt{\Delta}\log\Delta \cdot h)$. In the case that the noise is uniform across nodes, these bounds become $2(1-\varepsilon)^h$ and $\mathcal{O}(\sqrt{\Delta}\cdot h)$ respectively. The constant hidden in the $\mathcal{O}$ notation only depends polynomially on $1/\varepsilon$.

Taking $h = -3\log(2n)/\log(1-\varepsilon)$, gives $\mathbb{P}(u \text{ is misleading}) \leq 1/n^2$. Denote by $\mathtt{Good}$ the event that none of the separators encountered are misleading. By a union bound, $\mathbb{P}(\mathtt{Good}^{\,c}) \leq 1/n$.

$$\mathcal{Q}(\mathtt{A}_{sep}) = \mathbb{P}(\mathtt{Good})\,\mathcal{Q}(\mathtt{A}_{sep} \mid \mathtt{Good}) + \mathbb{P}(\mathtt{Good}^{\,c})\,\mathcal{Q}(\mathtt{A}_{sep} \mid \mathtt{Good}^{\,c}). \tag{3.1}$$

As $\mathtt{A}_{sep}$ runs an exhaustive search algorithm in parallel, the second term is $\mathcal{O}(1)$. For the first term, note that conditioning on $\mathtt{Good}$, all local procedures either find the treasure or give the correct answer, and so there are $\mathcal{O}(\log n)$ of them and they eventually find the treasure. Denote by u_i the i-th vertex that $\mathtt{local}_h$ is executed on. By linearity of expectation, and applying Lemma 3.5, the first term of (3.1) is $\mathbb{P}(\mathtt{Good})\sum_i \mathcal{Q}(\mathtt{local}_h(u_i) \mid \mathtt{Good}) = \mathcal{O}(\log n \cdot \sqrt{\Delta}\log\Delta \cdot h) = \mathcal{O}(\sqrt{\Delta}\log\Delta\log^2 n)$. As $\log(1+x) > x$ always, then $-1/\log(1-\varepsilon) \leq 1/\varepsilon$, and the hidden factor in the $\mathcal{O}$ is as stated. □

3.5.1 Proof of Lemma 3.5

The proof makes use of Lemma 2.8, proven in the previous chapter. To check the probability that u is misleading, consider two cases:

1. $\tau \notin T_h(u)$, and τ_u is not promising. By Lemma 2.8, and recalling that $\Delta^h \leq \beta(\tau_u)$, the

probability τ_u is not promising is:

$$\mathbb{P}\left(\sum_{w\in[u,\tau_u\rangle} -X_w \le \frac{2}{3}h\log(\Delta)\right) = \mathbb{P}\left(\sum_{w\in[u,\tau_u\rangle} X_w \ge -\frac{2}{3}\cdot h\log(\Delta)\right)$$
$$\le \prod_{w\in[u,\tau_u\rangle} \frac{1-\varepsilon}{\sqrt{\Delta_w}}\cdot e^{\frac{3}{4}\cdot\frac{2}{3}h\log(\Delta)} = \frac{(1-\varepsilon)^{d(u,\tau_u)}}{\sqrt{\beta(\tau_u)}}\Delta^{\frac{h}{2}} \le (1-\varepsilon)^{d(u,\tau_u)}.$$

As $d(u,\tau_u) \ge \log_\Delta \beta(\tau_u) \ge h$, this is at most $(1-\varepsilon)^h$.

2. If v is a nominee that is not in the same connected component of $T\setminus\{u\}$ as τ_u, then by Lemma 2.8, the probability that v is promising is

$$\mathbb{P}\left(\sum_{w\in[u,v\rangle} X_w \ge \frac{2}{3}\log\Delta\cdot h\right) \le \prod_{w\in[u,v\rangle}\frac{1-\varepsilon}{\sqrt{\Delta_w}}\cdot e^{-\frac{3}{4}\cdot\frac{2}{3}h\log\Delta}$$
$$= \frac{(1-\varepsilon)^{d(u,v)}}{\sqrt{\beta(v)}}\Delta^{-\frac{h}{2}} \le \frac{(1-\varepsilon)^{d(u,v)}}{\Delta^h}.$$

However, denote by L the set of nominees in T_u. As they are a subset of the leaves of T_u, by the way θ is defined:

$$1 \ge \sum_{x\in L}\theta(v) \ge \sum_{x\in L}\frac{1}{\beta(v)} \ge \sum_{x\in L}\frac{1}{\Delta^{h+1}} = \frac{|L|}{\Delta^{h+1}} \tag{3.2}$$

So, $|L| \le \Delta^{h+1}$. Therefore, by a union bound, the probability that there exists a nominee v that renders u misleading is at most $\Delta(1-\varepsilon)^h$.

The probability that u is misleading is then at most $(1+\Delta)(1-\varepsilon)^h$ as stated. In the case where the noise parameter does not depend on each node, the analysis is the same, except that in (3.2), $\beta(v) = \Delta^h$, and so following the same logic, $|L| \le \Delta^h$, and this part contributes only $(1-\varepsilon)^h$.

For the second part of the lemma, consider some event X where u is not misleading. As τ_u is either the actual treasure or promising, and acts as the treasure in the eyes of $\mathtt{A}_{walk}$, then the local procedure stops when it encounters τ_u. It might actually stop before (because it found another promising node), so,

$$\mathbb{P}(X)\,\mathcal{Q}(\mathtt{local}(u)\mid X) \le \mathbb{P}(X)\,\mathcal{Q}(\mathtt{A}_{walk}(T_h(u))\mid X)$$
$$\le \mathcal{Q}(\mathtt{A}_{walk}(T_h(u))) = \mathcal{O}(\sqrt{\Delta}\cdot\mathtt{depth}(T_h(u)))$$

But the depth of T_u is at most $\mathcal{O}(h\log\Delta)$, since its leaves satisfy $\beta(v) < \Delta^{h+1}$, and $\beta(v) \ge 2^{\mathtt{depth}(v)}$. For the case of a regular tree, $\beta(v) = \Delta^{\mathtt{depth}(v)}$ and so the depth of T_u is at most h, giving the result.

3.6 Proof of Corollary 3.3

As explained in the introduction, this proof builds on the proof of Theorem 3.2 and uses the same terminology. It also relies on Theorem 2.5 presented in the previous section.

We use a local procedure described in Lemma 3.6 that allows us to learn with probability $1 - O(\frac{\delta}{\log n})$, in which one of the connected components of $T \setminus \{u\}$ the treasure resides. By a union bound, applying this local procedure on a separator of the tree, and recursing on the component pointed out by the procedure, allows to find the treasure in logarithmic number of runs of the local procedure with probability at least $1 - O(\delta)$.

Lemma 3.6

Let $d \in \mathbb{N}$ be big enough. Let u be a separator. There exists a search procedure that queries the vicinity of u up to distance d and outputs either $(*)$ the component of $T \setminus \{u\}$ that contains τ or $(**)$ τ itself if $d(\tau, u) \leq d$. The success probability is at least $1 - \delta$ and the number of queries is not greater than $(\delta^{-1}d)^{O(\varepsilon^{-1})}$.

Proof. Let T_u be the subtree of depth d rooted at u. Hereafter, the reference tree is T_u, so the notion of fitness is with respect to T_u (that is, the depth parameter involved in Definition 2.22 is d and not D).

Let us first describe the promised local procedure. It consists in applying algorithm A_{walk} from Theorem 2.5 on T_u until either finding the treasure or finding a reachable fit node at distance precisely d from u, denoted x. We will see that with high probability at least one of these events holds so that the behavior of the local procedure when none of these events happen is not relevant. For the sake of concreteness, we could say that it stops if all of T_u has been explored. We note that Algorithm A_{walk} makes walking steps, which are viewed as queries in the query model in this context. The output of the local search procedure is either τ, if it was found, or the component of x in $T \setminus \{u\}$.

Let us analyse the performance of the local search procedure. If $\tau \in T_u$, then Theorem 2.5 ensures that it is found with probability $1-\delta$ in at most $(\delta^{-1}d)^{O(\varepsilon^{-1})}$ steps. Otherwise, consider the node x at distance d from u in T_u that is on the path to τ. Within T_u, the advice is sampled as if x was the treasure τ, so Theorem 2.5 guarantees in this case that x is found with probability $1 - \delta$ in less than $(\delta^{-1}d)^{O(\varepsilon^{-1})}$ steps. Moreover, under that event x is fit and reachable.

To complete the argument, we just need to guarantee that with high probability, there are no reachable fit nodes at distance d from u outside the component of x. Then, the local procedure may discover a reachable and fit node at distance d different from x, but it will still be in the same component as x.

Recall that a 0-node is a node whose common ancestor with the treasure is the root (which is u in this case, since the reference tree is T_u). These are precisely the nodes in other components than the component of x. Claim 2.26 asserts that all reachable fit 0-nodes are within distance $h_2(d)$ of u with probability at least $1 - \frac{\delta}{4}$. We write $h_2(d)$ to emphasize that the parameter is defined here as a function of d, namely $\frac{6}{\varepsilon^2}\log_\Delta(4\delta^{-1}d)$. We see that if d is big enough (as a

function of ε, Δ) then $h_2(d) < d$.

Hence with that probability, any fit node at distance d found by the local procedure is guaranteed to be in the right component, that is the component to which x belongs.

Overall, the success probability of this procedure is $1 - \delta - \frac{\delta}{4} = 1 - \frac{5}{4}\delta$. We may write $\delta' = \frac{4}{5}\delta$ and get the desired statement[1]. □

To conclude, we apply Lemma 3.6 with $\delta' = \delta/\log n$ and $d = \log n$ for every one of the at most $\log n$ separators leading to τ. By a union bound, the success probability is $1 - \delta$ and the total number of queries is $\log n \cdot (\delta^{-1} \log n)^{O(\varepsilon^{-1})} = (\delta^{-1} \log n)^{O(\varepsilon^{-1})}$.

Letting h be the depth of the local search procedure, no separator on the way to the treasure is misleading with probability $> 1 - O(\log n e^{-ch})$ for some parameter c that depends on ε only. Under this event, all local search procedures find the correct way to proceed. Each of these local search procedures takes at most $h^{O(\varepsilon^{-1})}$ moving steps (and hence queries) with probability $1 - h^{-3}$. We use here Theorem 2.5 with an enhanced probability guarantee.

Overall, setting $h = c \log n$ for some large constant $c > 0$, we obtain that with probability greater than $1 - O(\log n \ \cdot e^{-ch}) - \log n h^{-3} > 1 - O(\log^{-1} n)$ ends correctly at takes at most $(\log n)^{O(\varepsilon^{-1})}$ steps and this concludes the proof.

3.7 Proof of Theorem 3.1

In this section, we present algorithm $\mathtt{A}_{2-\text{layers}}$ that performs almost optimally (up to lower order terms) in the regime where $q < c/\sqrt{\Delta}$ for some small enough positive constant c (as opposed to $q < (1-\varepsilon)\Delta^{-1/2}$ as in Theorem 3.2). More precisely, in that regime, it finds the treasure in $\mathcal{O}(\sqrt{\Delta} \log n \cdot \log\log n)$ queries in expectation. Moreover, in contrast to Theorem 3.2, in this section we do not allow the mistake the mistake parameter to depend on the node.

Before we continue, let us note that taking a small enough c, the condition $q < c/\sqrt{\Delta}$ we are using here actually implies[2] Condition ($\star$) with $\varepsilon = (1 - 2^{-1/4})/2$.

Algorithm $\mathtt{A}_{2-\text{layers}}$ runs two algorithms in parallel, namely, $\mathtt{A}_{fast}$, and $\mathtt{A}_{mid}$. Algorithm $\mathtt{A}_{fast}$ is actually $\mathtt{A}_{sep}$, except that it applies the local procedure with parameter h being $h_2 = \lceil \kappa_2 \log\log n \rceil$ rather than $\Theta(\log n)$. Algorithm $\mathtt{A}_{mid}$ is similar to $\mathtt{A}_{sep}$, as it also uses h being $h_1 = \lceil \kappa_1 \log n \rceil$. However it uses a different local exploration procedure, see more details in Section 3.7.1. κ_1 and κ_2 are constant independent of n whose value will be determined later. We will henceforth omit the ceiling $\lceil \cdot \rceil$ in the interest of readability.

Let us first recall some of the definitions that were introduced in Section 3.5. Here $T_h(u)$ denotes the tree of nodes at distance at most h from u. Call a leaf $v \in T_h(u)$ a *nominee* if

[1] In fact, the rescaling $\delta' = \frac{4}{5}\delta$ could be avoided, by observing that the event of probability $1 - \delta/4$ discussed in the proof is included in the $1 - \delta$ probability event that guarantees the success of A_{walk}. This follows from inspecting the proof of Theorem 2.5.

[2] Indeed, recall that for regular trees, Condition ($\star$) reads $q < \frac{1-\varepsilon-\Delta^{-1/4}}{\sqrt{\Delta}+\Delta^{1/4}}$. Now, $\Delta \geq 2$ implies that $1 - \Delta^{-1/4} \geq 1 - 2^{-1/4}$ and $\Delta^{1/4} \leq \sqrt{\Delta}$. Hence $\frac{1-\varepsilon-\Delta_v^{-1/4}}{\sqrt{\Delta}+\Delta^{1/4}} \geq \frac{1-2^{-1/4}-\varepsilon}{2}\frac{1}{\sqrt{\Delta}}$. We may set $\varepsilon = \frac{1-2^{-1/4}}{2}$ so that, as soon as $c < \frac{1-2^{-1/4}-\varepsilon}{2} = \frac{1-2^{-1/4}}{4}$, $q < c\Delta^{-1/2}$ implies Condition ($\star$) with that choice of ε.

its distance to u is exactly h. Denote by $\mathcal{U}(u)$ the set of nominees that are not in the same component as τ_u in $T \setminus \{u\}$. Call a nominee *promising* if $\sum_{w \in [u,v\rangle} X_w \geq \frac{2}{3}h$, where $X_w = 1$ if the advice at w is pointing to v, $X_w = -1$ if it is pointing to u, and $X_w = 0$ otherwise. Note that X_u can never be -1. Let τ_u be the leaf on $T_h(u)$ closest to τ if $\tau \notin T_h(u)$ and $\tau_u = \tau$ otherwise. Recall also that u is called *h-misleading*, if one of the two following happens (1) $\tau_u \neq \tau$ and τ_u is not promising, or (2) There is some promising nominee in $\mathcal{U}(u)$.

Let `Excellent` be the event that no separator on the way to the treasure is h_2-misleading. The following claim is a direct consequence of Lemma 3.5 (regular tree case) and linearity of expectation, summing the query complexity of the $\lceil \log n \rceil$ separators on the way to the treasure.

Claim 3.7

$$\mathbb{P}(\texttt{Excellent}) \cdot \mathcal{Q}(\mathtt{A}_{fast} \mid \texttt{Excellent}) = \mathcal{O}\left(\sqrt{\Delta} \log n \cdot \log\log n\right).$$

To bound the total expected number of queries, we run in parallel algorithm $\mathtt{A}_{mid}$. All that remains is then to prove that $\mathbb{P}(\texttt{Excellent}^c) \cdot \mathcal{Q}(\mathtt{A}_{mid} \mid \texttt{Excellent}^c) = \mathcal{O}(\sqrt{\Delta} \log n)$.

3.7.1 Algorithm $\mathtt{A}_{mid}$

As mentioned, $\mathtt{A}_{mid}$ is similar to $\mathtt{A}_{sep}$ except that it uses a different local procedure. More precisely, recall that $\mathtt{A}_{sep}$ executes Procedure $\mathtt{local}_h(u)$ by running $\mathtt{A}_{walk}$ on $T_h(u)$ until it either finds the treasure or finds a promising nominee, and in the latter case, it declares that the treasure is on the connected component of $T \setminus \{u\}$ containing this nominee. In the context of Algorithm $\mathtt{A}_{mid}$, for technical commodity, we choose to run Procedure $\mathtt{local}_h(u)$ with a simpler exploration routine which we call $\mathtt{A}_{loop}$. It is less efficient than $\mathtt{A}_{walk}$ but its simplicity will be useful for analyzing its behaviour in various, "less clean", circumstances. Indeed, we will need to analyse the performances of $\mathtt{A}_{loop}$, conditioning on the event $\texttt{Excellent}^c$, implying that some parts of the tree have to be pointing in the wrong direction.

The fact that $\mathtt{A}_{loop}$ is less efficient than $\mathtt{A}_{walk}$ will not affect the final bound, as its running time will dominate the total running time with very low probability.

Algorithm $\mathtt{A}_{loop}$. Recall in this section we only deal with Δ-regular trees. Define *level* i as the set of all nodes at distance i from the root. At each round, $\mathtt{A}_{loop}$ only compares nodes within a given level i. Specifically, it goes to the node in level i with most arrows pointing at it among the non-visited nodes in level i. It only considers vertices whose parent has been explored already. The index i is incremented modulo the depth of the tree D, on every round. Below is a description in pseudocode. The loop over i explains the name $\mathtt{A}_{loop}$.

In what follows we will analyse Algorithm $\mathtt{A}_{loop}$ conditioning on some parts of the tree being misleading. For readability considerations, the interested reader might wish to first see how it behaves on a simpler scenario, without any conditioning. The proof of this is shown in the full version of the paper (see [31]).

Algorithm 1: Algorithm $\mathtt{A}_{loop}$

1 Continuously loop over levels $1, 2, \ldots, D$
2 When considering level i, go to the yet unexplored reachable node at the current level (if one exists) that has most arrows pointing to it.

Lemma 3.8

Consider a (not necessarily complete) Δ-ary tree. Then $\mathcal{Q}(\mathtt{A}_{loop}) = \mathcal{O}(D^3\sqrt{\Delta})$.

In fact, a slightly more refined analysis shows that $\mathcal{Q}(\mathtt{A}_{loop}) = \mathcal{O}(D^2\sqrt{\Delta})$, but this not needed for our current purposes, and so we omit it.

3.7.2 Analysis of $\mathtt{A}_{mid}$ Conditioning On The Complement of Excellent

To complete the proof of Theorem 3.1 we will show that if c small enough, then

$$\mathbb{P}(\mathtt{Excellent}^{\,c}) \cdot \mathcal{Q}\left(\mathtt{A}_{mid} \mid \mathtt{Excellent}^{\,c}\right) = \mathcal{O}(\sqrt{\Delta}\log n).$$

Decomposing $\mathtt{Excellent}^{\,c}$. At a high level, we seek to break $\mathtt{Excellent}^{\,c}$ into many elementary bad events. Denote $u_1, \ldots u_\ell$ the sequence of separators on the way to the treasure τ. Note that $\ell \leq \lceil \log n \rceil$. First,

$$\mathtt{Excellent}^{\,c} = \bigcup_{i \leq \ell} \{u_i \text{ is } h_2\text{-misleading}\}.$$

Using the union bound argument in Section 3.8 (Claim 3.13),

$$\mathcal{Q}\left(\mathtt{A}_{mid} \bigcap \mathtt{Excellent}^{\,c}\right) \leq \sum_{i \leq \ell} \mathcal{Q}\left(\mathtt{A}_{mid} \bigcap u_i \text{ is } h_2\text{-misleading}\right), \tag{3.3}$$

where, to keep the equation light we write $\mathcal{Q}(\mathtt{A} \bigcap E)$ in place of $\mathcal{Q}(\mathtt{A} \mid E) \cdot \mathbb{P}(E)$ where $\mathtt{A}$ is an algorithm and E is an event.

Since we ultimately want to show that the left hand side in the previous equation is $\mathcal{O}(\sqrt{\Delta}\log n)$, it is sufficient to show that for any fixed $i \leq \ell$,

$$\mathcal{Q}\left(\mathtt{A}_{mid} \bigcap u_i \text{ is } h_2\text{-misleading}\right) = \mathcal{O}(\sqrt{\Delta}). \tag{3.4}$$

From now on, we fix i and focus on the case where u_i is h_2-misleading. Recall that algorithm $\mathtt{A}_{mid}$, just as $\mathtt{A}_{sep}$, proceeds in phases of local exploration, running also an exhaustive search in parallel to handle the case that one of the local explorations ends with a wrong answer. Denote by $\mathtt{Good}$ the event that all separators on the way to the treasure, namely, $u_1, \ldots, u_\ell$, are not

h_1-misleading. Under $\texttt{Good}$, the local exploration phases amount to running $\mathtt{A}_{loop}$ on $T_{h_1}(u_j)$ for $j \leq \ell$. Now,

$$\mathcal{Q}\left(\mathtt{A}_{mid} \bigcap u_i \text{ is } h_2\text{-misleading}\right) = \mathcal{Q}\left(\mathtt{A}_{mid} \bigcap (u_i \text{ is } h_2\text{-misleading} \cap \texttt{Good }\right) + \mathcal{Q}\left(\mathtt{A}_{mid} \bigcap (u_i \text{ is } h_2\text{-misleading} \cap \neg\texttt{Good }\right).$$

By Lemma 3.5 (regular tree case),

$$\mathbb{P}(\neg\texttt{Good }) \leq 2(1-\varepsilon)^{h_1} = 2(1-\varepsilon)^{\kappa_1 \log n}.$$

Recall that Condition $(\star)$ is satisfied with the constant $\varepsilon = (1-2^{-1/4})/2$, and so taking κ_1 to be a large enough constant, gives that $\mathbb{P}(\neg\texttt{Good }) > 1/n$. This means that if $\texttt{Good}$ does not hold, it is fine to resort to exhaustive search, as the second term above becomes $\mathcal{O}(1)$. Also, since $\mathtt{A}_{mid}$ runs $\mathtt{A}_{loop}$ on local subtrees until it finds a promising nominee, and conditioned on $\texttt{Good}$, the local "treasure" is such a promising nominee, then the number of queries made by such a local run is bounded above by number of queries $\mathtt{A}_{loop}$ needs to find the treasure there. So by linearity of expectation,

$$\begin{aligned}
&\mathcal{Q}\left(\mathtt{A}_{mid} \bigcap u_i \text{ is } h_2\text{-misleading}\right) \\
&\leq \sum_{j \leq \log n} \mathcal{Q}\left(\mathtt{A}_{loop}\left(T_{h_1}(u_j)\right) \bigcap (u_i \text{ is } h_2\text{-misleading} \cap \texttt{Good }\right) + \mathcal{O}(1) \\
&\leq \sum_{j \leq \log n} \mathcal{Q}\left(\mathtt{A}_{loop}\left(T_{h_1}(u_j)\right) \bigcap u_i \text{ is } h_2\text{-misleading}\right) + \mathcal{O}(1).
\end{aligned}$$

The last inequality follows from the fact that for any algorithm A and any two events $E_1 \subseteq E_2$, $\mathcal{Q}(A \bigcap E_1) \leq \mathcal{Q}(A \bigcap E_2)$.

For the sake of lightening notations, we henceforth refer to u_j as σ' and u_i as u. This choice of notations reflects the fact that we are rooting the tree at $u_j = \sigma'$ and running $\mathtt{A}_{loop}$ on $T_{h_1}(\sigma')$. The fact that σ' and u are separators is not relevant in this analysis. We also denote by τ_u the leaf on $T_{h_2}(u)$ that is closest to τ and by τ' the leaf of $T_{h_1}(u)$ that is closest to τ or simply τ if $\tau \in T_{h_1}(u)$. With these notations Equation (3.4) immediately follows once we prove:

Lemma 3.9

For any $\sigma', u \in T$,

$$\mathcal{Q}\left(\mathtt{A}_{loop}\left(T_{h_1}(\sigma')\right) \bigcap u \text{ is } h_2\text{-misleading}\right) = \mathcal{O}\left(\frac{\sqrt{\Delta}}{\log n}\right).$$

Decomposing the event $\{u$ is h_2-misleading$\}$. So far we saw that it is sufficient to analyse the events where one separator is h_2-misleading. We now pursue decomposing these events into even smaller ones. To this aim the following definition is convenient.

Definition 3.10

Let $a, b \in T$ be two nodes such that a is the closest one to τ out of the nodes in $[a, b]$. Noting that a vertex can never point to itself:

- For $S \subseteq \langle a, b]$, denote by $M^S_{\text{sides}}(a, b)$ the event that the nodes of S neither point towards a nor towards b.
- For $S \subseteq [a, b\rangle$, denote by $M^S_{\text{up}}(a, b)$ the event that the nodes of S all point towards b.

Claim 3.11

For any a, b and S as in Definition 3.10,

- $\mathbb{P}\left(M^S_{\text{sides}}(a, b)\right) \leq q^{|S|}$,
- $\mathbb{P}\left(M^S_{\text{up}}(a, b)\right) \leq \left(\frac{q}{\Delta}\right)^{|S|}$.

Let us now see in more detail what it means for a node u to be h_2-misleading. First recall from the definition that $|[u, \tau_u]| = h_2$, as otherwise $\tau \in T_{h_2}(u)$ and u cannot be h_2-misleading because of the path $[u, \tau_u]$. Several cases need to be considered.

1. τ_u is not promising, and so the sum of advice on $[u, \tau_u\rangle$ is strictly less than $\frac{2}{3}h_2$. In this case, at least one of the following two must be true:
   (a) There are $\frac{1}{6}h_2$ locations on the path $[u, \tau_u\rangle$ where the advice points outside of the path (the value of the corresponding X_i's is 0). This corresponds to $M^S_{\text{sides}}(\tau_u, u)$ for some set $S \subseteq [u, \tau_u\rangle$ of size[3] $|S| = \frac{1}{6}h_2$.
   (b) There are $\frac{1}{12}h_2$ locations on $\langle u, \tau_u\rangle$ that point towards u (the value of the corresponding X_i's is 1). This corresponds to $M^S_{\text{up}}(\tau_u, u)$ for some set $S \subseteq [u, \tau_u]$ of size $|S| = \frac{1}{12}h_2$.
2. Some $v \in \mathcal{U}(u)$ is promising. In this case there must be some $\frac{2}{3}h_2$ locations on $[u, v]$ that point towards v. This corresponds to $M^S_{\text{up}}(u, v)$ for some $S \subseteq M^S_{\text{up}}([v, u])$ of size $|S| = \frac{2}{3}h_2$.

Define $\mathcal{C}(u) = \{S \subseteq [u, \tau_u] \mid |S| = \frac{1}{6}h_2\}$ and $\mathcal{D}(u) = \{S \subseteq [u, \tau_u] \mid |S| = \frac{1}{12}h_2\}$. Similarly define $\mathcal{E}(u) = \{(v, S) \mid v \in \mathcal{U}(u),\ S \subseteq [u, v],\ \text{and } |S| = \frac{2}{3}h_2\}$. Combining Definition 3.10 with

[3] Here again we omit the $\lceil \cdot \rceil$.

the previous paragraph, yields

$$\{u \text{ is } h_2\text{-misleading}\} \subseteq \{\tau_u \text{ is not promising}\} \cup \bigcup_{v \in \mathcal{U}(u)} \{v \text{ is promising}\}$$
$$\subseteq \bigcup_{S \in \mathcal{C}(u)} M^S_{\text{sides}}(\tau_u, u) \bigcup_{S \in \mathcal{D}(u)} M^S_{\text{up}}(\tau_u, u) \bigcup_{(v,S) \in \mathcal{E}(u)} M^S_{\text{up}}(u, v).$$

In fact, $\mathcal{E}(u)$ needs to be further decomposed. For each $v \in \mathcal{E}(u)$, let $k(v) = |[u, v] \cap [\sigma', \tau']|$. For each non-negative integer $k \geq 0$, let

$$\mathcal{E}_k(u) = \{(v, S) \in \mathcal{E}(u) \mid k(v) = k\}.$$

Clearly, $\mathcal{E}(u) = \cup_{k=0}^{h_2} \mathcal{E}_k(u)$.

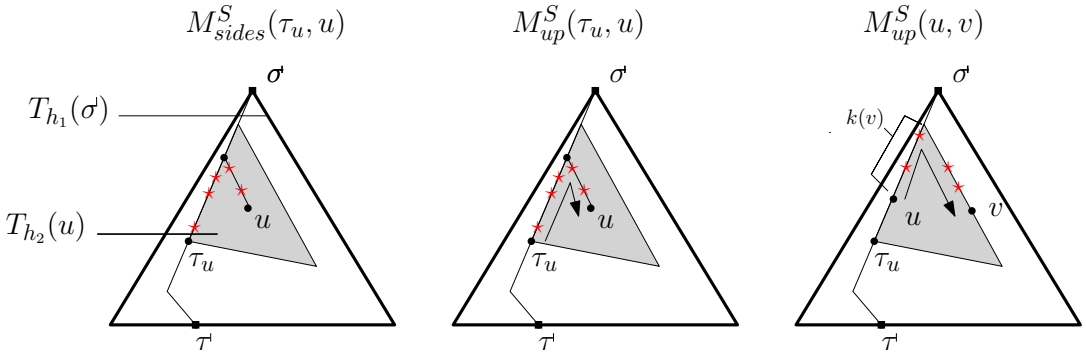

Figure 3.2: Different relative positions of u, τ_u and σ'. The path $[u, \tau_u]$ and different mistake patterns. In the left one mistakes (depicted as red stars) point outside of $[u, \tau_u]$, in the second they point towards u and in the third towards a nominee of $T_{h_2}(u)$, $v \in \mathcal{U}(u)$.

Using the union bound (Claim 3.13) as in Equation 3.3, the aforementioned decomposition implies:

$$\mathcal{Q}\left(\mathtt{A}_{loop}\left(T_{h_1}(\sigma')\right) \bigcap u \text{ is } h_2\text{-misleading}\right) \leq \sum_{S \in \mathcal{C}(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{sides}}(\tau_u, u)\right)$$
$$+ \sum_{S \in \mathcal{D}(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{up}}(\tau_u, u)\right)$$
$$+ \sum_{k=0}^{h_2} \sum_{(v,S) \in \mathcal{E}_k(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{up}}(u, v)\right) \tag{3.5}$$

To prove Lemma 3.9, our goal will be to show that each sum in the above equation is at most $\mathcal{O}(\sqrt{\Delta}/\log n)$.

3.7.3 Analysing Atomic Expressions

To prove that each sum is indeed $\mathcal{O}(\sqrt{\Delta}/\log n)$ we use the following two lemmas (proved in Section 3.7.4), which encapsulate the core of this proof, namely, the resilience of $\mathtt{A}_{loop}$ to certain kinds of error patterns.

Lemma 3.7.1. *Consider a tree T rooted at σ with treasure located at τ. Let $a, b \in T$ be two nodes such that a is the closest one to τ out of the nodes in $[a, b]$. Then,*

$$\mathcal{Q}\left(\mathtt{A}_{loop} \mid M^S_{sides}(a, b)\right) = \mathcal{O}\left(D^4 \Delta^{\frac{|S|+1}{2}}\right).$$

Lemma 3.7.2. *Consider a tree T rooted at σ with treasure located at τ. Let $a, b \in T$ be two nodes such that a is the closest one to τ out of the nodes in $[a, b]$. Then,*

$$\mathcal{Q}\left(\mathtt{A}_{loop} \mid M^S_{up}(a, b)\right) = \mathcal{O}\left(D^4 \Delta^{K+\frac{1}{2}} 4^{|S|}\right),$$

where $K = |S \cap [\sigma, \tau]|$.

As a first step to bounding the three sums of Equation (3.5), note that:

$$|\mathcal{C}(u)| \leq 2^{h_2} \tag{3.6}$$

$$|\mathcal{D}(u)| \leq 2^{h_2}, \tag{3.7}$$

$$|\mathcal{E}_k(u)| \leq 2^{h_2} \Delta^{h_2 - k}. \tag{3.8}$$

Indeed, $\mathcal{C}(u), \mathcal{D}(u)$ are sets of subsets of a path of length h_2. For the last term, the number of $v \in \mathcal{U}(u)$ at distance h_2 from u for which $k(v) = k$ is bounded by $\Delta^{h_2 - k}$. Now the three sums:

1. $S \in \mathcal{C}(u)$, so $S \subseteq [u, \tau_u]$ and $|S| = \frac{1}{6} h_2$, and τ_u is the closest to τ of all the nodes on the path. By Lemma 3.7.1,

   $$\mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \mid M^S_{\text{sides}}(\tau_u, u)\right) = \mathcal{O}\left(h_1^4 \Delta^{\frac{|S|+1}{2}}\right).$$

   According to Claim 3.11,

   $$\mathbb{P}(M^S_{\text{sides}}(\tau_u, u)) \leq q^{|S|}.$$

   Combining these bounds and (3.6) yields

   $$\begin{aligned}\sum_{S \in \mathcal{C}(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{sides}}(\tau_u, u)\right) &= \mathcal{O}\left(2^{h_2} \cdot q^{|S|} \cdot h_1^4 \Delta^{\frac{|S|+1}{2}}\right) \\ &= \mathcal{O}\left(\sqrt{\Delta} \cdot 2^{h_2} \cdot c^{|S|} \cdot h_1^4\right),\end{aligned}$$

   because $q < c/\sqrt{\Delta}$. Recall that $h_1 = \kappa_1 \log n$, $h_2 = \kappa_2 \log\log n$, and $|S| = \frac{1}{6} h_2$. κ_1 was already set to be some constant. Taking a large enough κ_2 and a small enough c, both independent of n, the previous expression is $\mathcal{O}(\sqrt{\Delta}/\log n)$ as needed.

2. $S \in \mathcal{D}(u)$, so $S \subseteq [u, \tau_u]$ and $|S| = \frac{1}{12}h_2$. Therefore, by Lemma 3.7.2,

$$\mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \mid M^S_{\text{up}}(\tau_u, u)\right) = \mathcal{O}\left(h_1^4 \Delta^{|S|+\frac{1}{2}} 2^{h_2}\right).$$

Because $K \leq |S|$ and $4^{|S|} \leq 2^{h_2}$. Combined with Claim 3.11 and (3.7):

$$\begin{aligned}\sum_{S \in \mathcal{D}(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{up}}(\tau_u, u)\right) &= \mathcal{O}\left(2^{h_2} \cdot \left(\frac{q}{\Delta}\right)^{|S|} \cdot h_1^4 \Delta^{|S|+\frac{1}{2}} 2^{h_2}\right) \\ &= \mathcal{O}\left(\sqrt{\Delta} \cdot 4^{h_2} \cdot q^{|S|} h_1^4\right)\end{aligned}$$

Again, since $|S| = \frac{1}{12}h_2$, then c and κ_2 can be chosen so that this is $\mathcal{O}(\sqrt{\Delta}/\log n)$.

3. $(v, S) \in \mathcal{E}_k(u)$, where $v \in \mathcal{U}(u)$, $S \subseteq [u, v]$, and $|S| = \frac{2}{3}h_2$. Also, $|[u, v] \cap [\sigma', \tau']| = k$, and so $|S \cap [\sigma', \tau']| \leq k$. As $v \in \mathcal{U}(u)$, then u is the closest to treasure of the vertices on $[u, v]$. By Lemma 3.7.2,

$$\mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \mid M^S_{\text{up}}(u, v)\right) = \mathcal{O}\left(h_1^4 \Delta^{k+\frac{1}{2}} 4^{h_2}\right)$$

Combined with (3.8) and Claim 3.11:

$$\begin{aligned}&\sum_{k=0}^{h_2} \sum_{(v,S) \in \mathcal{E}_k(u)} \mathcal{Q}\left(\mathtt{A}_{loop}(T_{h_1}(\sigma')) \bigcap M^S_{\text{up}}(u, v)\right) \\ &= \mathcal{O}\left(\sum_{k \leq h_2} 2^{h_2} \Delta^{h_2 - k} \cdot \left(\frac{q}{\Delta}\right)^{\frac{2}{3}h_2} h_1^4 \cdot \Delta^{k+\frac{1}{2}} 4^{h_2}\right) \\ &= \mathcal{O}\left(\sqrt{\Delta} \cdot h_2 8^{h_2} h_1^4 \left(q^2 \Delta\right)^{\frac{1}{3}h_2}\right). \\ &= \mathcal{O}\left(\sqrt{\Delta} \cdot h_2 8^{h_2} h_1^4 \cdot c^{\frac{1}{3}h_2}\right).\end{aligned}$$

Similarly to the two previous sums, this whole expression can be made as small as $\mathcal{O}(\sqrt{\Delta}/\log n)$.

Note that we assumed for simplicity that u, τ_u and v are all inside $T_{h_1}(\sigma')$. If they are not, we take nodes that are the closest to them on this subtree, which can only improve the bounds.

This concludes the proof of Lemma 3.9 and hence completes the proof of Theorem 3.1.

3.7.4 The Lemmas About the Resilience of $\mathtt{A}_{loop}$

Lemma 3.7.1 ***(restated).*** *Consider a tree T rooted at σ with treasure located at τ. Let $a, b \in T$ be two nodes such that a is the closest one to τ out of the nodes in $[a, b]$. Then,*

$$\mathcal{Q}\left(\mathtt{A}_{loop} \mid M^S_{sides}(a, b)\right) = \mathcal{O}\left(D^4 \Delta^{\frac{|S|+1}{2}}\right).$$

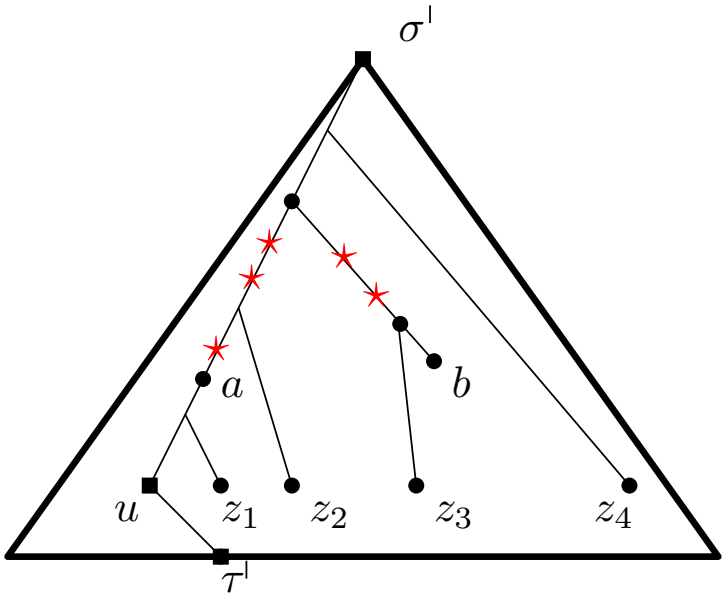

Figure 3.3: Notations introduced in the proof of Lemma 3.7.1 and 3.7.2. Points of S are depicted in red. On the figure $n(z_1) = 0, m(z_1) = 0$, $n(z_2) = 1, m(z_2) = 0$, $n(z_3) = 3, m(z_3) = 2$ and $n(z_4) = 3, m(z_4) = 0$.

Proof. As in the proof of Lemma 3.8, we break the number of queries made by $\mathtt{A}_{loop}$ conditioning on $M^S_{\text{sides}}(a, b)$ into a sum of random variables Q_j which correspond to the number of queries needed to discover the j-th node on the path $[\sigma, \tau]$ once the $(j-1)$-th was discovered. Each Q_j is bounded above by D times the expected number of competitors who beat this node. This is because each phase takes D steps, and only a subset of these nodes will actually be checked by $\mathtt{A}_{loop}$ on layer j before trying the correct node. Hence, a bound on the number of competitors who beat a given $u \in [\sigma, \tau]$ translates to a bound on $\mathcal{Q}\left(\mathtt{A}_{loop}(T) \mid M^S_{\text{sides}}(\tau, \sigma)\right)$ by multiplying it by D^2.

Let u be such a node, and z be a competitor of u (i.e., it is at the same level as u). Define $k(z)$ as half the distance between z and u, and denote $n(z) := |S \cap [\sigma, \tau] \cap [u, z]|$ and $m(z) := |S \cap [\sigma, \tau]^c \cap [u, z]|$. See Figure 3.3 for illustraion.

First note, that since all advice of $S \subseteq [a, b]$ points sideways w.r.t. to this path, then any of it which is on the path $[u, z]$ also points sideways w.r.t. it, except possibly at one point, which may actually point towards z. The different cases are seen in Figure 3.3:

- For z_1, the paths do not intersect at all.
- In the case of z_2, if the least common ancestor of u and z_2 was a member of S, then it could point towards z_2, and that would be sideways w.r.t. $[a, b]$.
- For z_3, the least common ancestor of b and z_3 could point towards z_3.

- For z_4, the least common ancestor of a and b could point towards z_4.
- There is also the case where $a \notin [\sigma, \tau]$, which is not depicted on Figure 3.3. The analysis remains valid, and in fact $n(z) = 0$ for all competitors z.

This one special vertex, if it exists, conditioned on that it points sideways w.r.t. $[a, b]$, points towards z with probabilty $1/(\Delta - 2)$, and otherwise points sideways w.r.t. $[u, z]$.

Fix k, n and m, and consider a competitor z such that $k(z) = k$, $n(z) = n$, and $m(z) = m$. On the path $[u, z]$ the number of advice remaining to be sampled is $2k - n - m - 1$. By Lemma 3.12:

$$\begin{aligned}
\mathbb{P}(z \text{ beats } u) &\leq \left(1 - \frac{1}{\Delta - 2}\right) \mathbb{P}\left(\sum_{s=1}^{2k-1-n-m} X_s \geq 0\right) + \frac{1}{\Delta - 2} \mathbb{P}\left(\sum_{s=1}^{2k-1-n-m} X_s \geq -1\right) \\
&= \left(\frac{1}{\sqrt{\Delta}}\right)^{2k-1-n-m} + \frac{4}{\Delta - 2} \left(\frac{1}{\sqrt{\Delta}}\right)^{2k-2-n-m} \\
&= \left(1 + \frac{4\sqrt{\Delta}}{\Delta - 2}\right) \left(\frac{1}{\sqrt{\Delta}}\right)^{2k-1-n-m} \leq 7 \cdot \left(\frac{1}{\sqrt{\Delta}}\right)^{2k-1-n-m},
\end{aligned}$$

as $\Delta \geq 3$. For fixed k, n, m there are at most Δ^{k-m} nodes z with $k(z) = k$ and $m(z) = m$. Also, for each such node, $n + m \leq 2k$. Hence, the total expected number of competitors that beat u is at most:

$$\sum_{k \leq D, n+m \leq 2k} \Delta^{k-m} \cdot 7 \left(\frac{1}{\sqrt{\Delta}}\right)^{2k-1-m-n}$$

For each choice of k there is exactly one corresponding value of n. This n satisfies $n \leq |S|$. There are also at most D choices for m. Thus, the above is at most

$$7 \cdot \sum_{k \leq D, n+m \leq 2k} \Delta^{(n+1-m)/2} = \mathcal{O}\left(D^2 \Delta^{(|S|+1)/2}\right).$$

□

Lemma 3.7.2 ***(restated).*** *Consider a tree T rooted at σ with treasure located at τ. Let $a, b \in T$ be two nodes such that a is the closest one to τ out of the nodes in $[a, b]$. Then,*

$$\mathcal{Q}\left(\mathtt{A}_{loop} \mid M_{up}^{S}(a, b)\right) = \mathcal{O}\left(D^4 \Delta^{K + \frac{1}{2}} 4^{|S|}\right),$$

where $K = |S \cap [\sigma, \tau]|$.

Proof. Let u be a node on the path $[\sigma, \tau]$. Our aim is to show that the expected number of competitors of u that beat it is $\mathcal{O}(D^2 \Delta^{K+\frac{1}{2}} 4^{|S|})$.

As in the proof of Lemma 3.7.1, let z be a competitor of u. Define $k(z)$ as half the distance between z and u, namely $k(z) := d(z,u)/2$. Denote $n(z) := |S \cap [\sigma,\tau] \cap [u,z]|$, and $m(z) := |S \cap [\sigma,\tau]^c \cap [u,z]|$.

Fixing k, n and m, take a competitor z such that $k(z) = k$, $n(z) = n$, and $k(z) = k$. The probability that such a z beats u is

$$\mathbb{P}\left(\sum_{s=1}^{2k-1-n-m} X_s \geq -n-m\right) \leq 4^{n+m}\Delta^{n+m-k+\frac{1}{2}},$$

by Lemma 3.12. There are at most Δ^{k-m} such nodes z. We bound the probability that each of these nodes z beats the treasure using the trivial bound 1 or the one above, depending on whether $n+m \leq k$ or $n+m > k$. Hence the total expected number of competitors of u who beat it is at most

$$\sum_{k\leq D, n+m\leq k} \Delta^{k-m} \cdot 4^{n+m}\Delta^{n+m-k+\frac{1}{2}} + \sum_{k\leq D, n+m>k} \Delta^{k-m}.$$

Since $n+m \leq |S|$, and $n \leq K$, the first term is at most:

$$4^{|S|} \sum_{k\leq D, n+m\leq k} \Delta^{K+\frac{1}{2}} \leq 4^{|S|} \cdot D^2 \cdot \Delta^{K+\frac{1}{2}},$$

where we used the fact that there are most D distinct values for k and D distinct values for m, while there is only one choice of n for each k. As for the second term, since $n+m > k$, then it is at most:

$$\sum_{k\leq D, n+m>k} \Delta^{n} \leq \sum_{k\leq D, n+m>k} \Delta^{K} \leq \sum_{k,m\leq D} \Delta^{K} \leq D^2 \cdot \Delta^{K},$$

concluding the proof. □

3.8 Complementary Proofs

3.8.1 Another Large Deviation Estimate

Here, we introduce another large deviation estimate used for the analysis of the query algorithm for regular trees. It gives better results for large h, yet works only for identical random variables, and so suits only the case of uniform noise, unlike Lemma 2.8.

Lemma 3.12

Consider random variables X_i taking values $\{-1, 0, 1\}$ with respective probabilities $(1-q, q\left(1-\frac{2}{\Delta}\right), \frac{q}{\Delta})$. If $q < \frac{c}{\sqrt{\Delta}}$ where $c < 1/64$, then for all $0 \le h \le l$,

$$\mathbb{P}\left(\sum_{i=1}^{\ell} X_i \ge -h\right) \le (4\sqrt{\Delta})^h \Delta^{-\ell/2}$$

Proof. Assume $\sum_{i=1}^{\ell} X_i \ge -h$. Denote by $j := \{i \mid X_i = 1\}$. As the number of -1's is at least h, then $j \le (\ell - h)/2$. There must also be at least $\ell - h - 2j$ zeros amongst what remains, otherwise the sum is less than $-h$. Using a union bound over the value of j and the locations of the ones and zeros we get:

$$\begin{aligned}
\mathbb{P}\left(\sum_{i=1}^{\ell} X_i \ge -h\right) &\le \sum_{j=0}^{\frac{\ell-h}{2}} \binom{\ell}{j}\binom{\ell-j}{\ell-h-2j}\left(q(1-\frac{2}{\Delta})\right)^{\ell-h-2j}\left(\frac{q}{\Delta}\right)^j \\
&\le \sum_{j=0}^{\frac{\ell-h}{2}} \binom{\ell}{j}\binom{\ell-j}{\ell-h-2j} q^{\ell-h-2j}\left(\frac{q}{\Delta}\right)^j \\
&\le 3^\ell \sum_{j=0}^{\frac{\ell-h}{2}} q^{\ell-h-2j}\left(\frac{q}{\Delta}\right)^j \le 3^\ell \cdot \frac{\ell}{2}\left(q^{\ell-h} + \left(\frac{q}{\Delta}\right)^{\frac{\ell-h}{2}}\right).
\end{aligned}$$

The last step uses the bound $\sum_{j=1}^{N} \rho^k \le N \cdot (\rho + \rho^N)$. Note that $x/2 < (4/3)^x$ always, and assigning $q < c/\sqrt{\Delta}$, this is at most:

$$4^\ell \frac{1}{\sqrt{\Delta}^{\ell-h}}\left(c^{\ell-h} + \left(\frac{\sqrt{c}}{\Delta^{3/2}}\right)^{\ell-h}\right) \le 4^\ell \frac{1}{\sqrt{\Delta}^{\ell-h}}\left(c^{\ell-h} + \sqrt{c}^{\ell-h}\right) \le 4^\ell \left(\frac{2\sqrt{c}}{\sqrt{\Delta}}\right)^{\ell-h}.$$

Since $c < 1/64$, then $2\sqrt{c} \le 1/4$ which means that this is at most

$$4^h \left(\frac{1}{\sqrt{\Delta}}\right)^{\ell-h},$$

giving the desired bound. □

3.8.2 Algorithm $\mathtt{A}_{loop}$ without Conditioning

Lemma (Lemma 3.8 restated)**.** *Consider a (not necessarily complete) Δ-ary tree. Then $\mathcal{Q}(\mathtt{A}_{loop}) = \mathcal{O}(D^3\sqrt{\Delta})$.*

Proof. Denote by $N_{layer}(u)$ the number of nodes on the same depth as u which have more discovered arrows than u pointing to them. This definition is central because of the following observation. The number of moves needed before finding u_{i+1} once u_i has been found is less than $\mathcal{O}(DN_{layer}(u_i))$. Indeed, once u_i is discovered, only *a subset of* the nodes which have more arrows pointing to them than u_{i+1} on layer $i+1$ are tried before u_{i+1} (at step (2) in the pseudocode description). The loop over the levels (at step (1)) induces a multiplicative factor of $\mathcal{O}(D)$.

Using linearity of expectation, it only remains to estimate $\mathbb{E}\left(N_{layer}(u_i)\right)$ where u_i is the ancestor of the treasure at depth $i \leq d$. There are at most Δ^ℓ nodes on layer i at distance $2\ell-1$ from u_i, for any $1 \leq \ell \leq i$. Moreover the probability that each of these nodes has more arrows pointing towards it than u_i exactly corresponds to $\mathbb{P}\left(\sum_{j=1}^{2\ell-1} X_j \geq 0\right)$, with the notations of Lemma 3.12.

Indeed, when comparing the amount of advice pointing to two different nodes u and v, only the nodes of $\langle u, v\rangle$ matter.

When estimating the probability that v beats u, each random variable X_j has to be interpreted as taking value $+1$ if the advice points towards v, -1 if it points towards u, and 0 if it points neither to u nor v. In the case that $u = u_j$ and v is another node on layer j, these events happen respectively with probability q/Δ, $1 - q + q/\Delta$ and $q(1 - 2\frac{1}{\Delta})$.

This means that for each i,

$$\mathbb{E}\left(N_{layer}(u_i)\right) \leq \sum_{\ell=1}^{i} \mathbb{P}\left(\sum_{j=1}^{2\ell-1} X_j \geq 0\right) \Delta^\ell \leq \sum_{\ell=1}^{d} \mathbb{P}\left(\sum_{j=1}^{2\ell-1} X_j \geq 0\right) \Delta^\ell.$$

By Lemma 3.12 this is at most

$$\mathcal{O}\left(\sum_{\ell=1}^{d} \Delta^{-\ell+\frac{1}{2}} \cdot \Delta^\ell\right) = \mathcal{O}(D\sqrt{\Delta}) = \mathcal{O}\left(D\sqrt{\Delta}\right).$$

□

3.8.3 Special Form of Union Bound

Claim 3.13

Let A be an event that can be decomposed as the union of events $(A_i)_{i\in I}$, $A \subseteq \bigcup_{i\in I} A_i$. Let X be a random variable.

$$\mathbb{E}(X \mid A)\mathbb{P}(A) \leq \sum_i \mathbb{E}(X \mid A_i)\mathbb{P}(A_i)$$

Proof. We denote by $\chi(B)$ the indicator function of event B. Then

$$\begin{aligned}\mathbb{E}(X \mid A)\mathbb{P}(A) = \mathbb{E}(X \cdot \chi(A)) &\leq \mathbb{E}\left(X \cdot \chi\left(\bigcup_i A_i\right)\right) \leq \mathbb{E}\left(X \cdot \sum_i \chi(A_i)\right) \\ &= \sum_i \mathbb{E}\left(X \cdot \chi(A_i)\right) = \sum_i \mathbb{E}(X \mid A_i)\mathbb{P}(A_i).\end{aligned}$$

Where we used the union bound in the form $\chi(\bigcup_i A_i) \leq \sum_i \chi A_i$ and then linearity of expectation. □

Chapter 4

Limits for Rumor Spreading in Stochastic Populations

4.1 Introduction

4.1.1 Background and motivation

Systems composed of tiny mobile components must function under conditions of unreliability. In particular, any sharing of information is inevitability subject to communication noise. The effects of communication noise in distributed living systems appears to be highly variable. While some systems disseminate information efficiently and reliably despite communication noise [67, 35, 99, 110], others generally refrain from acquiring social information, consequently losing all its potential benefits [76, 107, 114]. It is not well understood which characteristics of a distributed system are crucial in facilitating noise reduction strategies and, conversely, in which systems such strategies are bound to fail. Progress in this direction may be valuable towards better understanding the constraints that govern the evolution of cooperative biological systems.

Computation under noise has been extensively studied in the computer science community. These studies suggest that different forms of error correction (*e.g.,* redundancy) are highly useful in maintaining reliability despite noise [5, 116, 115]. All these, however, require the ability to transfer significant amount of information over stable communication channels. Similar redundancy methods may seem biologically plausible in systems that enjoy stable structures, such as brain tissues.

The impact of noise in stochastic systems with ephemeral connectivity patterns is far less understood. To study these, we focus on *rumor spreading* - a fundamental information dissemination task that is a prerequisite to almost any distributed system [23, 36, 45, 87]. A successful and efficient rumor spreading process is one in which a large group manages to quickly learn information initially held by one or a few informed individuals. Fast information flow to the whole group dictates that messages be relayed between individuals. It currently remains unclear what

are the precise conditions that enable fast rumor spreading. On the one hand, recent works indicate that in some models of random noisy interactions, a collective coordinated process can in fact achieve fast information spreading [63, 72]. These models, however, are based on *push* operations that inherently include a certain reliable component (see more details in Section 4.1.3). On the other hand, other works consider computation through noisy operations, and show that several distributed tasks require significant running time [77]. The tasks considered in these works (including the problem of learning the input bits of all processors, or computing the parity of all the inputs) were motivated by computer applications, and may be less relevant for biological contexts. Moreover, they appear to be more demanding than basic tasks, such as rumor spreading, and hence it is unclear how to relate bounds on the former problems to the latter ones.

In this chapter we take a general stance to identify limitations under which reliable and fast rumor spreading cannot be achieved. Modeling a well-mixed population, we consider a passive communication scheme in which information flow occurs as one agent observes the cues displayed by another. If these interactions are perfectly reliable, the population could achieve extremely fast rumor spreading [87]. In contrast, here we focus on the situation in which messages are noisy. Informally, our main theoretical result states that when all components of communication are noisy then fast rumor spreading through large populations is not feasible. In other words, our results imply that fast rumor spreading can only be achieved if either 1) the system exhibits some degree of structural stability or 2) some facet of the pairwise communication is immune to noise. In fact, our lower bounds hold even when individuals are granted unlimited computational power and even when the system can take advantage of complete synchronization.

Finally, we corroborate our theoretical findings with new analyses regarding the efficiency of information dissemination during recruitment by desert ants. More specifically, we analyze data from an experiment conducted at the Weizmann Institute of Science, concerning recruitment in *Cataglyphis niger* desert ants [106]. These analyses suggest that this biological system lacks reliability in all its communication components, and its deficient performances qualitatively validate our predictions. We stress that this approach is highly uncommon. Indeed, using empirical biological data to validate predictions from theoretical distributed computing is extremely rare. We believe, however, that this interdisciplinary methodology may carry significant potential, and hope that it will be useful for future works that will follow this framework.

4.1.2 The Problem

An intuitive description of the model follows. For more precise definitions, see Section 4.2.

Consider a population of n *agents*. Thought of as computing entities, assume that each agent has a discrete internal *state*, and can execute randomized algorithms - by internally flipping coins. In addition, each agent has an *opinion*, which we assume for simplicity to be binary, *i.e.,* either 0 or 1. A small number, s, of agents play the role of *sources*. Source agents

are aware of their role and share the same opinion, referred to as the *correct opinion*. The goal of all agents is to have their opinion coincide with the correct opinion.

To achieve this goal, each agent continuously displays one of several *messages* taken from some finite alphabet Σ. Agents interact according to a random pattern, termed as the *parallel-$\mathcal{PULL}$* model: In each round $t \in \mathbb{N}^+$, each agent u observes the message currently displayed by another agent v, chosen uniformly at random (u.a.r) from all agents. Importantly, communication is noisy, hence the message observed by u may differ from that displayed by v. The noise is characterized by a *noise parameter* $\delta > 0$. Our model encapsulates a large family of noise distributions, making our bounds highly general. Specifically, the noise distribution can take *any* form, as long as it satisfies the following criterion.

Definition 4.1.1 (The δ-uniform noise criterion)**.** *Any time some agent u observes an agent v holding some message $m \in \Sigma$, the probability that u actually receives a message m' is at least δ, for any $m' \in \Sigma$. All noisy samples are independent.*

When messages are noiseless, it is easy to see that the number of rounds that are required to guarantee that all agents hold the correct opinion with high probability is $\mathcal{O}(\log n)$ [87]. In what follows, we aim to show that when the δ-uniform noise criterion is satisfied, the number of rounds required until even one non-source agent can be moderately certain about the value of the correct opinion is very large. Specifically, thinking of δ and s as constants independent of the population size n, this time is at least $\Omega(n)$.

To prove the lower bound, we will bestow the agents with capabilities that far surpass those that are reasonable for biological entities. These include:

- Unique identities: Agents have unique identities in the range $\{1, 2, \ldots n\}$. When observing agent v, its identity is received without noise.
- Complete knowledge of the system: Agents have access to all parameters of the system (including n, s, and δ) as well as to the full knowledge of the initial configuration except, of course, the correct opinion and the identity of the sources. In addition, agents have access to the results of random coin flips used internally by all other agents.
- Full synchronization: Agents know when the execution starts, and can count rounds.

We show that even given this extra computational power, fast convergence cannot be achieved.

4.1.3 Our Contributions

Theoretical Results

In all the statements that follow we consider the parallel-$\mathcal{PULL}$ model satisfying the δ-uniform noise criterion, where $cs/n < \delta \leq 1/2$ for some sufficiently large constant c. Note that our criterion given in Definition 4.1.1 implies that $\delta \leq 1/|\Sigma|$. Hence, the previous lower bound on δ implies a restriction on the alphabet size, specifically, $|\Sigma| \leq n/(cs)$.

Theorem 4.1

Any rumor spreading protocol cannot converge in less than $\Omega(\frac{n\delta}{s^2(1-2\delta)^2})$ rounds.

Recall that a source is aware that it is a source, but if it wishes to identify itself as such to agents that observe it, it must encode this information in a message, which is, in turn, subject to noise. We also consider the case in which an agent can reliably identify a source when it observes one (i.e.,, this information is not noisy). For this case, the following bound, which is weaker than the previous one but still polynomial, apply (a formal proof appears in the full version of the paper):

Corollary 4.2

Assume that sources are reliably detectable. There is no rumor spreading protocol that converges in less than $\Omega((\frac{n\delta}{s^2(1-2\delta)^2})^{1/3})$ rounds.

Our results suggest that, in contrast to systems that enjoy stable connectivity, structureless systems are highly sensitive to communication noise. More concretely, the two crucial assumptions that make our lower bounds work are: 1) stochastic interactions, and 2) δ-uniform noise (see the right column of Figure 4.1). When agents can stabilize their interactions the first assumption is violated. In such cases, agents can overcome noise by employing simple error-correction techniques, *e.g.,* using redundant messaging or waiting for acknowledgment before proceeding. As demonstrated in Figure 4.1 (left column), when the noise is not uniform, it might be possible to overcome it with simple techniques based on using default neutral messages, and employing exceptional distinguishable signals only when necessary.

Exponential Separation Between $\mathcal{PUSH}$ and $\mathcal{PULL}$

Our lower bounds on the parallel-$\mathcal{PULL}$ model (where agents observe other agents) should be contrasted with known results in the parallel-$\mathcal{PUSH}$ model, which is the push equivalent to parallel-$\mathcal{PULL}$ model, where in each round each agent may or may not actively push a message to another agent chosen u.a.r. (see also Section 4.2.3). Although never proved, and although their combination is known to achieve more power than each of them separately [87], researchers often view the parallel-$\mathcal{PULL}$ and parallel-$\mathcal{PUSH}$ models as very similar on complete communication topologies. Our lower bound result, however, undermines this belief, proving that in the context of noisy communication, there is an exponential separation between the two models. Indeed, when the noise level is constant for instance, convergence (and in fact, a much stronger convergence than we consider here) can be achieved in the parallel-$\mathcal{PUSH}$ using only logarithmic number of rounds [63, 72], by a simple strategy composed of two stages. The first stage consists of providing all agents with a guess about the source's opinion, in such a way that ensures a non-negligible bias toward the correct guess. The second stage then boosts this bias by progressively amplifying it. A crucial aspect in the first stage is that agents

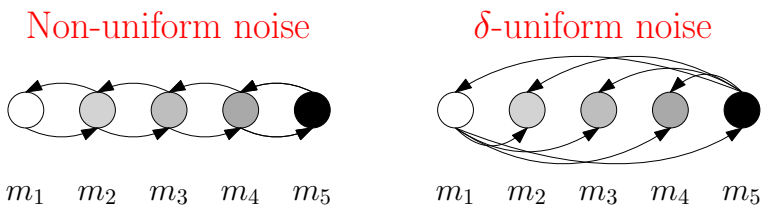

Figure 4.1: **Non-uniform noise vs. uniform noise.** On the left, we consider an example with non-uniform noise. Assume that the message vocabulary consists of 5 symbols, that is, $\Sigma = \{m_1, m_2, m_3, m_4, m_5\}$, where $m_1 = 0$ and $m_5 = 1$, represent the opinions. Assume that noise can occur only between consecutive messages. For example, m_2 can be observed as either m_2, m_3 or m_1, all with positive constant probability, but can never be viewed as m_4 or m_5. In this scenario, the population can quickly converge on the correct opinion by executing the following. The sources always display the correct opinion, *i.e.,* either m_1 or m_5, and each other agent displays m_3 unless it has seen either m_1 or m_5 in which case it adopts the opinion it saw and displays it. In other words, m_3 serves as a default message for non-source agents, and m_1 and m_5 serve as attracting sinks. It is easy to see that the correct opinion will propagate quickly through the system without disturbance, and within $\mathcal{O}(\log n)$ number of rounds, where n is the size of the population, all agents will hold it with high probability. In contrast, as depicted on the right picture, if every message can be observed as any other message with some constant positive probability (for clarity, some of the arrows have been omitted from the sketch), then convergence cannot be achieved in less than $\Omega(n)$ rounds, as Theorem 4.1 dictates.

remain silent until a certain point in time that they start sending messages continuously, which happens after being contacted for the first time. This prevents agents from starting to spread information before they have sufficiently reliable knowledge. It further allows to control the dynamics of the information spread in a balanced manner. More specifically, marking an edge corresponding to a message received for the first time by a node, the set of marked edges forms a spanning tree of low depth, rooted at the source. The depth of such tree can be interpreted as the deterioration of the message's reliability.

On the other hand, as shown here, in the parallel-$\mathcal{PULL}$ model, even with the synchonization assumption, rumor spreading cannot be achieved in less than a linear number of rounds. Perhaps the main reason why these two models are often considered similar is that with an extra bit in the message, a $\mathcal{PUSH}$ protocol can be *approximated* in the $\mathcal{PULL}$ model, by letting this bit indicate whether the agent in the $\mathcal{PUSH}$ model was aiming to push its message. However, for such a strategy to work, this extra bit has to be reliable. Yet, in the noisy $\mathcal{PULL}$ model, no bit is safe from noise, and hence, as we show, such an approximation cannot work. In this sense, the extra power that the noisy $\mathcal{PUSH}$ model gains over the noisy $\mathcal{PULL}$ model, is that the very fact that one node attempts to communicate with another is reliable. This, seemingly minor, difference carries significant consequences.

Generalizations

Several of the assumptions discussed earlier for the parallel-$\mathcal{PULL}$ model were made for the sake of simplicity of presentation. In fact, our results can be shown to hold under more general conditions, that include: 1) different rate for sampling a source, and 2) a more relaxed noise criterion. In addition, our theorems were stated with respect to the parallel-$\mathcal{PULL}$ model. In this model, at every round, each agent samples a single agent u.a.r. In fact, for any integer k, our analysis can be applied to the model in which, at every round, each agent observes k agents chosen u.a.r. In this case, the lower bound would simply reduce by a factor of k. Our analysis can also apply to a sequential variant, in which in each time step, two agents u and v are chosen u.a.r from the population and u observes v. In this case, our lower bounds would multiply by a factor of n, yielding, for example, a lower bound of $\Omega(n^2)$ in the case where δ and s are constants[1].

Recruitment in Desert Ants

Our theoretical results assert that efficient rumor spreading in large groups could not be achieved without some degree of communication reliability. An example of a biological system whose communication reliability appears to be deficient in all of its components is recruitment in *Cataglyphis niger* desert ants. In this species, when a forager locates an oversized food item, she returns to the nest to recruit other ants to help in its retrieval [6, 106].

We complement our theoretical findings by providing new analyses from an experiment on this system conducted at the Weizmann Institute of Science [106]. In such experimental setting, we interpret our theoretical findings as an abstraction of the interaction modes between ants. While such high-level approximation may be considered very crude, we retain that it constitutes a good trade-off between analytical tractability and experimental data.

In our experimental setup recruitment happens in the small area of the nest's entrance chamber (Figure 4.2a). We find that within this confined area, the interactions between ants are nearly uniform [102], such that an ant cannot control which of her nest mates she meets next (see Figure 4.2b). This random meeting pattern coincides with the first main assumption of our model. Additionally, it has been shown that recruitment in *Cataglyphis niger* ants relies on rudimentary alerting interactions [52, 85] which are subject to high levels of noise [106]. Furthermore, the responses to a recruiting ant and to an ant that is randomly moving in the nest are extremely similar [106]. Although this may resemble a noisy push interaction scheme, ants cannot reliably distinguish an ant that attempts to transmit information from any other non-communicating individual. In our theoretical framework, the latter fact means that the structure of communication is captured by a noisy-pull scheme (see more details about $\mathcal{PUSH}$ vs. $\mathcal{PULL}$ in Section 4.1.3).

It has previously been shown that the information an ant passes in an interaction can

[1] This increase in not surprising as each round in the parallel-$\mathcal{PULL}$ model consists of n observations, while the sequential model consists of only one observation in each time step.

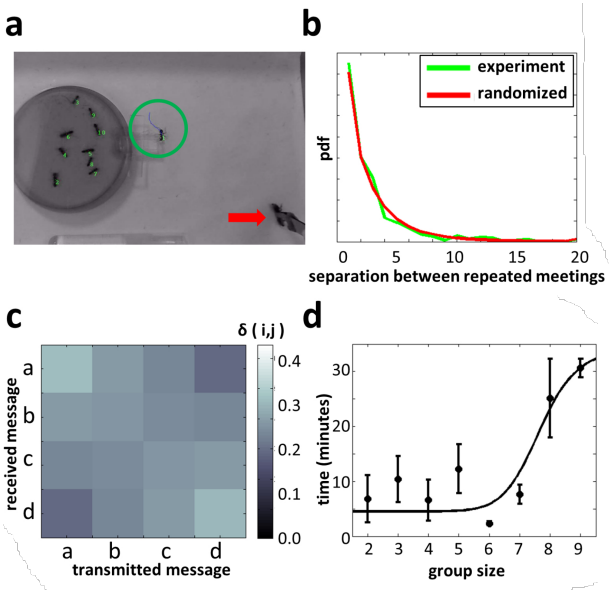

Figure 4.2: **Unreliable communication and slow recruitment by desert ant (*Cataglyphis niger*). a.** The experimental setup. The recruiter ant (circled) returns to the nest's entrance chamber (dark, 9cm diameter, disc) after finding the immobilized food item (arrow). Group size is ten. **b.** A *pdf* of the number of interactions that an ant experiences before meeting the same ant twice. The *pdf* is compared to uniform randomized interaction pattern. Data summarizes $N = 671$ interactions from seven experiments with a group size of 6 ants. **c.** Interactions with moving ants where classified into four different messages ('a' to 'd') depending on the ants' speed. The noise at which messages were confused with each other was estimated according to the response recipient, initially stationary, ants (see Materials and Methods). Gray scale indicates the estimated overlap between every two messages $\delta(i, j)$. Note that $\delta = \min(\delta(i, j)) \approx 0.2$. Data collected over $N = 64$ interactions. **d.** The mean time it takes an ant that is informed about the food to recruit two nest-mates to exit the nest is presented for two group size ranges.

be attributed solely to her speed before the interaction [106]. Binning ant speeds into four arbitrary discrete messages and measuring the responses of stationary ants to these messages, we can estimate the probabilities of one message to be mistakenly perceived as another one (see Materials and Methods). Indeed, we find that this communication is extremely noisy and complies with the uniform-noise assumption with a δ of approximately 0.2 (Figure 4.2c).

Given the coincidence between the communication patterns in this ant system and the requirements of our lower bound we expect long delays before any uninformed ant can be relatively certain that a recruitment process is occurring. We therefore measured the time it takes an ant, that has been at the food source, to recruit the help of two nest-mates. We find that this time increases with group size ($p < 0.05$ Kolmogorov-Smirnov test over $N = 24$ experiments, Figure 4.2d). Thus, in this system, inherently noisy interactions on the microscopic level have direct implications on group level performance. While group sizes in these experiments are small, we nevertheless find these recruitment times in accordance with our asymptotic theoretical results. More details on the experimental methodology can be found in the full version of the paper.

4.1.4 Related Work

Lower bound approaches in biological contexts are still extremely rare [22, 64]. Our approach can be framed within the general endeavour of addressing problems in theoretical biology through the algorithmic perspective of theoretical computer science [38, 37].

The computational study of abstract systems composed of simple individuals that interact using highly restricted and stochastic interactions has recently been gaining considerable attention in the community of theoretical computer science. Popular models include *population protocols* [12], which typically consider constant size individuals that interact in pairs (using constant size messages) in random communication patterns, and the *beeping* model [2], which assumes a fixed network with extremely restricted communication. Our model also falls in this framework as we consider the $\mathcal{PULL}$ model [45, 87, 89] with constant size messages. So far, despite interesting works that consider different fault-tolerant contexts [8, 9], most of the progress in this framework considered noiseless scenarios.

In *Rumor Spreading* problems (also referred to as *Broadcast*) a piece of information typically held by a single designated agent is to be disseminated to the rest of the population. It is the subject of a vast literature in theoretical computer science, and more specifically in the distributed computing community, see, *e.g.,* [23, 36, 45, 48, 63, 77, 87, 105]. While some works assume a fixed topology, the canonical setting does not assume a network. Instead agents communicate through uniform $\mathcal{PUSH}/\mathcal{PULL}$ based interactions (including the *phone call* model), in which agents interact in pairs with other agents independently chosen at each time step uniformly at random from all agents in the population. The success of such protocols is largely due to their inherent simplicity and fault-tolerant resilience [56, 87]. In particular, it has been shown that under the $\mathcal{PUSH}$ model, there exist efficient rumor spreading protocol that uses a single bit per message and can overcome flips in messages (noise) [63].

The line of research initiated by El-Gamal [44], also studies a broadcast problem with noisy interactions. The regime however is rather different from ours: all n agents hold a bit they wish to transmit to a single receiver. This line of research culminated in the $\Omega(n \log \log n)$ lower bound on the number of messages shown in [77], matching the upper bound shown many years sooner in [73].

4.1.5 Organization of the Chapter

The model was already briefly introduced above and in the Introduction of the manuscript. We give more details in Section 4.2. The proofs are presented in Section 4.3.

4.2 Formal Description of the Models

We consider a population of n agents that interact stochastically and aim to converge on a particular opinion held by few knowledgable individuals. For simplicity, we assume that the set of opinions contain two opinions only, namely, 0 and 1.

As detailed in this section, we shall assume that agents have access to significant amount of resources, often exceeding reasonable more realistic assumptions. Since we are concerned with lower bounds, we do not loose generality from such permissive assumptions. These liberal assumptions will actually simplify our proofs. One of these assumptions is the assumption that each agent is equipped with a unique identity $id(v)$ in the range $\{1, 2, \ldots, n\}$ (see more details in Section 4.2.4).

4.2.1 Initial Configuration

The initial configuration is described in several layers. First, the *neutral initial configuration* corresponds to the initial states of the agents, before the sources and the desired opinion to converge to are set. Then, a random initialization is applied to the given neutral initial configuration, which determines the set of sources and the opinion that agents need to converge to. This will result in what we call the *charged initial configuration*. It can represent, for example, an external event that was identified by few agents which now need to deliver their knowledge to the rest of the population.

Neutral Initial Configuration $\mathbf{x}^{(0)}$. Each agent v starts the execution with an *input* that contains, in addition to its identity, an initial *state* taken from some discrete set of states, and[2] a binary *opinion* variable $\lambda_v \in \{0, 1\}$. The *neutral initial configuration* $\mathbf{x}^{(0)}$ is the vector whose i'th index, $\mathbf{x}_i^{(0)}$ for $i \in [n]$, is the input of the agent with identity i.

Charged Initial Configuration and Correct Opinion. The charged initial configuration is determined in three stages. The first corresponds to the random selection of sources,

[2]The opinion of an agent could have been considered as part of the state of the agent. We separate these two notions merely for the presentation purposes.

the second to the selection of the correct opinion, and the third to a possible update of states of sources, as a result of being selected as sources with a particular opinion.

- **1st stage - Random selection of sources.** Given an integer $s \leq n$, a set S of size s is chosen uniformly at random (u.a.r) among the agents. The agents in S are called *sources*. Note that any agent has equal probability of being a source. We assume that each source knows it is a source, and conversely, each non-source knows it is not a source.
- **2nd stage - Random selection of correct opinion.** In the main model we consider, after sources have been determined in the first stage, the sources are randomly initialized with an opinion, called the *correct opinion*. That is, a fair coin is flipped to determine an opinion in $\{0, 1\}$ and all sources are assigned with this opinion.
- **3rd stage - Update of initial states of sources.** To capture a change in behavior as a result of being selected as a source with a particular opinion, we assume that once the opinion of a source u has been determined, the initial state of u may change according to some distribution $f_{source-state}$ that depends on (1) its identity, (2) its opinion, and (3) the neutral configuration. Each source samples its new state independently.

4.2.2 Alphabet and Noisy Messages

Agents communicate by observing each other according to some random pattern (for details see Section 4.2.3). To improve communication agents may choose which content, called *message*, they wish to reveal to other agents that observe them. Importantly, however, such messages are subject to noise. More specifically, at any given time, each agent v (including sources) displays a message $m \in \Sigma$, where Σ is some finite alphabet. The alphabet Σ agents use to communicate may be richer than the actual information content they seek to disseminate, namely, their opinions. This, for instance, gives them the possibility to express several levels of certainty [90]. We can safely assume that the size of Σ is at least two, and that Σ includes both symbols 0 and 1. We are mostly concerned with the case where Σ is of constant size (*i.e.,* independent of the number of agents), but note that our results hold for any size of the alphabet Σ, as long as the noise criterion is satisfied (see below).

δ-uniform noise. When an agent u *observes* some agent v, it receives a sample of the message currently held by v. More precisely, for any $m, m' \in \Sigma$, let $P_{m,m'}$ be the probability that, any time some agent u observes an agent v holding some message $m \in \Sigma$, u actually receives message m'. The probabilities $P_{m,m'}$ define the entries of the noise-matrix P [72], which does not depend on time. We hereby also emphasize that the agents' samples are independent.

The noise in the sample is characterized by a *noise parameter* $0 < \delta \leq 1/2$. One of the important aspects in our theorems is that they are general enough to hold assuming *any* distribution governing the noise, as long as it satisfies the following noise criterion.

Definition 4.2.1 (The noise uniformity parameter δ)**.** *We say that the noise has uniformity δ if $P_{m,m'} \geq \delta$ for any $m, m' \in \Sigma$.*

Observe that the aforementioned criterion implies that $\delta \leq 1/|\Sigma|$, and that the case $\delta = 1/|\Sigma|$ corresponds to messages being completely random, and the rumor spreading problem is thus unsolvable. We next define a weaker criterion, that is particularly meaningful in cases in which sources are more restricted in their message repertoire than general agents. This may be the case, for example, if sources always choose to display their opinion as their message (possibly together with some extra symbol indicating that they are sources). Formally, we define $\Sigma' \subseteq \Sigma$ as the set of possible messages that a source can hold together with the set of messages that can be observed when viewing a source (*i.e.,* after noise is applied). Our theorems actually apply to the following criterion, that requires that only messages in Σ' are attained due to noise with some sufficient probability.

Definition 4.2.2 (The relaxed noise uniformity parameter δ)**.** *We say that the noise has Σ'-relaxed uniformity δ if $P_{m,m'} \geq \delta$ for any $m \in \Sigma$ and $m' \in \Sigma'$.*

4.2.3 Random Interaction Patterns

We consider several basic interaction patterns. Our main model is the *parallel-$\mathcal{PULL}$* model. In this model, time is divided into *rounds*, where at each round $i \in \mathbb{N}^+$, each agent u independently selects an agent v (possibly $u = v$) u.a.r from the population and then u observes the message held by v. The *parallel-$\mathcal{PULL}$* model should be contrasted with the *parallel-$\mathcal{PUSH}$* model, in which u can choose between *sending* a message to the selected node v or doing nothing. We shall also consider the following variants of $\mathcal{PULL}$ model.

- *parallel-$\mathcal{PULL}$*(k). Generalizing *parallel-$\mathcal{PULL}$* for an integer $1 \leq k \leq n$, the *parallel-$\mathcal{PULL}$*(k) model allows agents to observe k other agents in each round. That is, at each round $i \in \mathbb{N}^+$, each agent independently selects a set of k agents (possibly including itself) u.a.r from the population and observes each of them.
- *sequential-$\mathcal{PULL}$*. In each time step $t \in \mathbb{N}^+$, two agents u and v are selected uniformly at random (u.a.r) among the population, and agent u observes v.
- *broadcast-$\mathcal{PULL}$*. It each time step $t \in \mathbb{N}^+$ one agent is chosen u.a.r. from the population and all agents observe it, receiving the same noisy sample of its message[3].

Regarding the difference in time units between the models, since interactions occur in parallel in the *parallel-$\mathcal{PULL}$* model, one round in that model should informally be thought of as roughly n time steps in the *sequential-$\mathcal{PULL}$* or *broadcast-$\mathcal{PULL}$* model.

4.2.4 Liberal Assumptions

As mentioned, we shall assume that agents have abilities that surpass their realistic ones. These assumption not only increases the generality of our lower bounds, but also simplifies

[3]The *broadcast-$\mathcal{PULL}$* model is mainly used for technical considerations. We use it in our proofs as it simplifies our arguments while not harming their generality. Nevertheless, this broadcast model can also capture some situations in which agents can be seen simultaneously by many other agents, where the fact that all agents observe the same sample can be viewed as noise being originated by the observed agent.

their proofs. Specifically, the following liberal assumptions are considered.

- **Unique identities.** Each agent is equipped with a unique identity $id(v) \in \{1, 2, \ldots, n\}$, that is, for every two agents u and v, we have $id(u) \neq id(v)$. Moreover, whenever an agent u observes some agent v, we assume that u can infer the identity of v. In other words, we provide agents with the ability to reliably distinguish between different agents at no cost.
- **Unlimited internal computational power.** We allow agents to have unlimited computational abilities including infinite memory capacity. Therefore, agents can potentially perform arbitrarily complex computations based on their knowledge (and their id).
- **Complete knowledge of the system.** Informally, we assume that agents have access to the complete description of the system except for who are the sources and what is their opinion. More formally, we assume that each agent has access to:
  - the neutral initial configuration $\mathbf{x}^{(0)}$,
  - all the systems parameters, including the number of agents n, the noise parameter δ, the number of sources s, and the distribution $f_{source-state}$ governing the update the states of sources in the third stage of the charged initial configuration.
- **Full synchronization.** We assume that all agents are equipped with clocks that can count time steps (in *sequential-*$\mathcal{PULL}$ or *broadcast-*$\mathcal{PULL}$) or rounds (in *parallel-*$\mathcal{PULL}$(k)). The clocks are synchronized, ticking at the same pace, and initialized to 0 at the beginning of the execution. This means, in particular, that if they wish, the agents can actually share a notion of time that is incremented at each time step.
- **Shared randomness.** We assume that algorithms can be randomized. That is, to determine the next action, agents can internally toss coins and base their decision on the outcome of these coin tosses. Being liberal, we shall assume that randomness is shared in the following sense. At the outset, an arbitrarily long sequence r of random bits is generated and the very same sequence r is written in each agent's memory before the protocol execution starts. Each agent can then deterministically choose (depending on its state) which random bits in r to use as the outcome of its own random bits. This implies that, for example, two agents can possibly make use of the very same random bits or merely observe the outcome of the random bits used by the other agents. Note that the above implies that, conditioning on an agent u being a non-source agent, all the random bits used by u during the execution are accessible to all other agents.
- **Coordinated sources.** Even though non-source agents do not know who the sources are, we assume that sources do know who are the other sources. This means, in particular, that the sources can coordinate their actions.

4.2.5 Considered Algorithms and Solution Concept

Upon observation, each agent can alter its internal state (and in particular, its message to be seen by others) as well as its opinion. The strategy in which agents update these variables is called "algorithm". As mentioned, algorithms can be randomized, that is, to determine the

next action, agents can use the outcome of coin tosses in the sequence r (see *Shared randomness* in Section 4.2.4). Overall, the action of an agent u at time t depends on:

1. the initial state of u in the charged initial configuration (including the identity of u and whether or not it is a source),
2. the initial knowledge of u (including the system's parameters and neutral configuration),
3. the time step t, and the list of its observations (history) up to time $t-1$, denoted $x_u^{(<t)}$,
4. the sequence of random bits r.

4.2.6 Convergence and Time Complexity

At any time, the opinion of an agent can be viewed as a binary *guess* function that is used to express its most knowledgeable guess of the correct opinion. The agents aim to minimize the probability that they fail to guess this opinion. In this context, it can be shown that the optimal guessing function is deterministic.

Definition 4.2.3. *We say that* convergence *has been achieved if one can specify a particular non-source agent v, for which is it guaranteed that its opinion is the correct opinion with probability at least* $2/3$*. The* time complexity *is the number of time steps (respectively, rounds) required to achieve convergence.*

We remark that the latter definition encompasses all three models considered.

Remark 4.2.4 (Different sampling rates of sources)**.** *We consider sources as agents in the population but remark that they can also be thought of as representing the environment. In this case, one may consider a different rate for sampling a source (environment) vs. sampling a typical agent. For example, the probability to observe any given source (or environment) may be x times more than the probability to observe any given non-source agent. This scenario can also be captured by a slight adaptation of our analysis. When x is an integer, we can alternatively obtain such a generalization by considering additional* artificial *sources in the system. Specifically, we replace each source u_i with a set of sources U_i consisting of x sources that coordinate their actions and behave identically, simulating the original behavior of u_i. (Recall that we assume that sources know who are the other sources and can coordinate their actions.) Since the number of sources increases by a multiplicative factor of x, our lower bounds (see Theorem 4.3 and Corollary 4.2) decrease by a multiplicative factor of x^2.*

4.3 The Lower Bounds

Throughout this section we consider $\delta < 1/2$, such that $\frac{(1-2\delta)}{\delta sn} \leq \frac{1}{10}$. Our goal in this section is to prove the following result.

Theorem 4.3

Assume that the relaxed δ-uniform noise criterion is satisfied.

- Let k be an integer. Any rumor spreading protocol on the *parallel-*$\mathcal{PULL}$(k) model cannot converge in fewer rounds than $\Omega\left(\frac{n\delta}{ks^2(1-2\delta)^2}\right)$.
- Consider either the *sequential-*$\mathcal{PULL}$ or the *broadcast-*$\mathcal{PULL}$ model. Any rumor spreading protocol cannot converge in fewer rounds than $\Omega\left(\frac{n^2\delta}{s^2(1-2\delta)^2}\right)$.

To prove the theorem, we first prove (in Section 4.3.1) that an efficient rumor spreading algorithm in either the noisy *sequential-*$\mathcal{PULL}$ model or the *parallel-*$\mathcal{PULL}$(k) model can be used to construct an efficient algorithm in the *broadcast-*$\mathcal{PULL}$ model. The resulted algorithm has the same time complexity as the original one in the context of *sequential-*$\mathcal{PULL}$ and adds a multiplicative factor of kn in the context of *parallel-*$\mathcal{PULL}$(k).

We then show how to relate the rumor spreading problem in *broadcast-*$\mathcal{PULL}$ to a statistical inference test (Section 4.3.2). A lower bound on the latter setting is then achieved by adapting techniques from mathematical statistics (Section 4.3.3).

4.3.1 Reducing to the *broadcast-*$\mathcal{PULL}$ Model

The following lemma establishes a formal relation between the convergence times of the models we consider. We assume all models are subject to the same noise distribution.

Lemma 4.4

Any protocol operating in *sequential-*$\mathcal{PULL}$ can be simulated by a protocol operating in *broadcast-*$\mathcal{PULL}$ with the same time complexity. Moreover, for any integer $1 \leq k \leq n$, any protocol $\mathcal{P}$ operating in *parallel-*$\mathcal{PULL}$(k) can be simulated by a protocol operating in *broadcast-*$\mathcal{PULL}$ with a time complexity that is kn times that of $\mathcal{P}$ in *parallel-*$\mathcal{PULL}$(k).

Proof. Let us first show how to simulate a time step of *sequential-*$\mathcal{PULL}$ in the *broadcast-*$\mathcal{PULL}$ model. Recall that in *broadcast-*$\mathcal{PULL}$, in each time step, all agents receive the same observation sampled u.a.r from the population. Upon drawing such an observation, all agents use their shared randomness to generate a (shared) uniform random integer X between 1 and n. Then, the agent whose unique identity corresponds to X is the one processing the observation, while all other agents ignore it. This reduces the situation to a scenario in *sequential-*$\mathcal{PULL}$, and the agents can safely execute the original algorithm designed for that model.

As for simulating a time step of *parallel-*$\mathcal{PULL}$(k) in *broadcast-*$\mathcal{PULL}$, agents divide time steps in the latter model into *rounds*, each composing of exactly kn time steps. Recall that the model assumes that agents share clocks that start when the execution starts and tick at each time step. This implies that the agents can agree on the division of time into rounds, and can

further agree on the round number. For $1 \leq i \leq kn$, during the i-th step of each round, only the agent whose identity is $(i \bmod n)+1$ receives[4] the observation, while all other agents ignore it. This ensures that when a round is completed in the *broadcast-$\mathcal{PULL}$* model, each agent receives exactly k independent uniform samples as it would in a round of *parallel-$\mathcal{PULL}$*(k). Therefore, at the end of each round $j \in \mathbb{N}^+$ in the *broadcast-$\mathcal{PULL}$* model, all agents can safely execute their actions in the j'th round of the original protocol designed for *parallel-$\mathcal{PULL}$*(k). This draws a precise bijection from rounds in *parallel-$\mathcal{PULL}$*(k) and rounds in *broadcast-$\mathcal{PULL}$*. The multiplicative overhead of kn simply follows from the fact that each round in *broadcast-$\mathcal{PULL}$* consists of kn time steps. □

Thanks to Lemma 4.4, Theorem 4.3 directly follows from the next theorem.

Theorem 4.5

Consider the *broadcast-$\mathcal{PULL}$* model and assume that the relaxed δ-uniform noise criterion is satisfied. Any rumor spreading protocol cannot converges in fewer time steps than $\Omega\left(\frac{n^2\delta}{s^2(1-2\delta)^2}\right)$.

The remaining of the section is dedicated to proving Theorem 4.5. Towards achieving this, we view the task of guessing the correct opinion in the *broadcast-$\mathcal{PULL}$* model, given access to noisy samples, within the more general framework of distinguishing between two types of stochastic processes which obey some specific assumptions.

4.3.2 Rumor Spreading and Hypothesis Testing

To establish the desired lower bound, we next show how the rumor spreading problem in the broadcast-$\mathcal{PULL}$ model relates to a statistical inference test. That is, from the perspective of a given agent, the rumor spreading problem can be understood as the following: Based on a sequence of noisy observations, the agent should be able to tell whether the correct opinion is 0 or 1. We formulate this problem as a specific task of distinguishing between two random processes, one originated by running the protocol assuming the correct opinion is 0 and the other assuming it is 1.

One of the main difficulties lies in the stochastic dependencies affecting these processes. In general, at different time steps, they do not consist of independent draws of a given random variable. In other words, the law of an observation not only depends on the correct opinion, on the initial configuration and on the underlying randomness used by agents, but also on the previous noisy observation samples and (consequently) on the messages agents themselves choose to display on that round. An intuitive version of this problem is the task of distinguishing between two (multi-valued) biased coins, whose bias changes according to the previous outcomes

[4]Receiving the observation doesn't imply that the agent processes this observation. In fact, it will store it in its memory until the round is completed, and process it only then.

of tossing them (*e.g.,* due to wear). Following such intuition, we define the following general class of *Adaptive Coin Distinguishing Tasks*, for short ACDT.

Definition 4.3.1 (ACDT)**.** *A* distinguisher *is presented with a sequence of observations taken from a coin of type η where $\eta \in \{0,1\}$. The type η is initially set to 0 or 1 with probability 1/2 (independently of everything else). The goal of the distinguisher is to determine the type η, based on the observations. More specifically, for a given time step t, denote the sequence of previous observations (up to, and including, time $t-1$) by $x^{(<t)} = (x^{(1)}, \ldots, x^{(t-1)})$. At each time t, given the type $\eta \in \{0,1\}$ and the history of previous observations $x^{(<t)}$, the distinguisher receives an observation $X_\eta^{(t)} \in \Sigma$, which has law*[5] *$P(X_\eta^{(t)} = m \mid x^{(<t)})$.*

We next introduce, for each $m \in \Sigma$, the parameter $\varepsilon(m, x^{(<t)}) = P(X_1^{(t)} = m \mid x^{(<t)}) - P(X_0^{(t)} = m \mid x^{(<t)})$. Since, at all times t, it holds that $\sum_{m\in\Sigma} P(X_0^{(t)} = m \mid x^{(<t)}) = \sum_{m\in\Sigma} P(X_1^{(t)} = m \mid x^{(<t)}) = 1$, then $\sum_{m\in\Sigma} \varepsilon(m, x^{(<t)}) = 0$. We shall be interested in the quantity $d_\varepsilon(x^{(<t)}) := \sum_{m\in\Sigma} |\varepsilon(m, x^{(<t)})|$, which corresponds to the ℓ_1 distance between the distributions $P(X_0^{(t)} = m \mid x^{(<t)})$ and $P(X_1^{(t)} = m \mid x^{(<t)})$ given the sequence of previous observations.

Definition 4.3.2 (The bounded family ACDT(ε, δ))**.** *We consider a family of instances of* ACDT*, called* ACDT(ε, δ)*, governed by parameters ε and δ. Specifically, this family contains all instances of* ACDT *such that for every t, and every history $x^{(<t)}$, we have:*

- *$d_\varepsilon(x^{(<t)}) \le \varepsilon$, and*
- *$\forall m \in \Sigma$ such that $\varepsilon(m, x^{(<t)}) \neq 0$, we have $\delta \le P(X_\eta^{(t)} = m \mid x^{(<t)})$ for $\eta \in \{0,1\}$.*

In the rest of the section, we show how Theorem 4.5, that deals with the *broadcast-$\mathcal{PULL}$* model, follows directly from the next theorem that concerns the adaptive coin distinguishing task, by setting $\varepsilon = \frac{2s(1-2\delta)}{n}$. The actual proof of Theorem 4.6 appears in Section 4.3.3.

Theorem 4.6

Consider any protocol for any instance of ACDT(ε, δ), The number of samples required to distinguish between a process of type 0 and a process of type 1 with probability of error less than $\frac{1}{3}$ is at least $\frac{\ln 2}{9}\left(\frac{6(\delta-\varepsilon)^3}{\delta^3-\delta^2\varepsilon+3\delta\varepsilon^2-\varepsilon^3}\right)\frac{\delta}{\varepsilon^2}$. In particular, if $\frac{\varepsilon}{\delta} < 10$, then the number of necessary samples is $\Omega\left(\frac{\delta}{\varepsilon^2}\right)$.

[5] We follow the common practice to use uppercase letters to denote random variables and lowercase letter to denote a particular realisation, e.g., $X^{(\le t)}$ for the sequence of observations up to time t, and $x^{(\le t)}$ for a corresponding realization.

Proof of Theorem 4.5 assuming Theorem 4.6

Consider a rumor spreading protocol $\mathcal{P}$ in the *broadcast-$\mathcal{PULL}$* model. Fix a node u. We first show that running $\mathcal{P}$ by all agents, the perspective of node u corresponds to a specific instance of $\texttt{ACDT}(\frac{2\texttt{s}(1-2\delta)}{\texttt{n}}, \delta)$ called $\Pi(\mathcal{P}, u)$. We break down the proof of such correspondence into two claims.

The ACDT instance $\Pi(\mathcal{P}, u)$. Recall that we assume that each agent knows the complete neutral initial configuration, the number of sources s, and the shared of random bits sequence r. We avoid writing such parameters as explicit arguments to $\Pi(\mathcal{P}, u)$ in order to simplify notation, however, we stress that what follows assumes that these parameters are fixed. The bounds we show hold for any fixed value of r and hence also when r is randomized.

Each agent is interested in discriminating between two families of charged initial configurations: Those in which the correct opinion is 0 and those in which it is 1 (each of these possibilities occurs with probability $\frac{1}{2}$). Recall that the correct opinion is determined in the 2nd stage of the charged initial configuration, and is independent form the choice of sources (1st stage).

We next consider the perspective of a generic non-source agent u, and define the instance $\Pi(\mathcal{P}, u)$ as follows. Given the history $x^{(<t)}$, we set $P(X_\eta^{(t)} = m \mid x^{(<t)})$, for $\eta \in \{0, 1\}$, to be equal to the probability that u observes message $m \in \Sigma$ at time step t of the execution $\mathcal{P}$. For clarity's sake, we remark that the latter probability is conditional on: the history of observations being $x^{(<t)}$, ,the sequence of random bits r, ,the correct opinion being $\eta \in \{0, 1\}$, ,the neutral initial configuration, ,the identity of u, ,the algorithm $\mathcal{P}$, and the system's parameters (including the distribution $f_{source-state}$ and the number of sources s).

Claim 4.7

Let $\mathcal{P}$ be a correct protocol for the rumor spreading problem in *broadcast-$\mathcal{PULL}$* and let u be an agent for which the protocol is guaranteed to produce the correct opinion with probability at least p by some time T (if one exists), for any fixed constant $p \in (0, 1)$. Then $\Pi(\mathcal{P}, u)$ can be solved in time T with correctness being guaranteed with probability at least p.

Proof. Conditioning on $\eta \in \{0, 1\}$ and on the random seed r, the distribution of observations in the $\Pi(\mathcal{P}, u)$ instance follows the distribution of observations as perceived from the perspective of u in *broadcast-$\mathcal{PULL}$*. Hence, if the protocol $\mathcal{P}$ at u terminates with output $j \in \{0, 1\}$ at round T, after the T-th observation in $\Pi(\mathcal{P}, u)$ we can set $\Pi(\mathcal{P}, u)$'s output to j as well. Given that the two stochastic processes have the same law, the correctness guarantees are the same. □

Lemma 4.8

$\Pi(\mathcal{P}, u) \in \texttt{ACDT}\left(\frac{2(1-2\delta)\mathtt{s}}{\mathtt{n}}, \delta\right)$.

Proof. Since the noise in *broadcast-$\mathcal{PULL}$* flips each message $m \in \Sigma$ into any $m' \in \Sigma'$ with probability at least δ, regardless of the previous history and of $\eta \in \{0,1\}$, at all times t, if $m \in \Sigma'$ then $P(X_\eta^{(t)} = m \mid x^{(<t)}) \geq \delta$. Consider a message $m \in \Sigma \setminus \Sigma'$ (if such a message exists). By definition, such a message could only be received by observing a non-source agent. But given the same history $x^{(<t)}$, the same sequence of random bits r, and the same initial knowledge, the behavior of a non-source agent is the same, no matter what is the correct opinion η. Hence, for $m \in \Sigma \setminus \Sigma'$ we have $P(X_0^{(t)} = m \mid x^{(<t)}) = P(X_1^{(t)} = m \mid x^{(<t)})$, or in other words, $m \in \Sigma \setminus \Sigma' \implies \varepsilon(m, x^{(<t)}) = 0$.

It remains to show that $d_\varepsilon(x^{(<t)}) \leq \frac{2(1-2\delta)s}{n}$. Let us consider two executions of the rumor spreading protocol, with the same neutral initial configuration, same shared sequence of random bits r, same set of sources, except that in the first the correct opinion is 0 while in the other it is 1. Let us condition on the history of observations $x^{(<t)}$ being the same in both processes. As mentioned, given the same history $x^{(<t)}$, the behavior of a non-source agent is the same, regardless of the correct opinion η. It follows that the difference in the probability of observing any given message is only due to the event that a source is observed. Recall that the number of sources is s. Therefore, the probability of observing a source is s/n, and we may write as a first approximation $\varepsilon(m, x^{(<t)}) \leq s/n$. However, we can be more precise. In fact, $\varepsilon(m, x^{(<t)})$ is slightly smaller than s/n, because the noise can still affect the message of a source. We may interpret $\varepsilon(m, x^{(<t)})$ as the following difference. For a source $v \in S$, let m_η^v be the message of u assuming the given history $x^{(<t)}$ and that v is of type $\eta \in \{0,1\}$ (the message m_η^v is deterministically determined given the sequence r of random bits, the neutral initial configuration, the parameters of the system, and the identity of v). Let $\alpha_{m',m}$ be the probability that the noise transforms a message m' into a message m. Then $\varepsilon(m, x^{(<t)}) = \frac{1}{n}\sum_{v\in S}(\alpha_{m_1^v,m} - \alpha_{m_0^v,m})$, and

$$d_\varepsilon(x^{(<t)}) = \sum_{m\in\Sigma} |\varepsilon(m, x^{(<t)})| \leq \frac{1}{n}\sum_{m\in\Sigma}\sum_{v\in S} |\alpha_{m_1^v,m} - \alpha_{m_0^v,m}|. \tag{4.1}$$

By the definition of $\texttt{ACDT}(\varepsilon, \delta)$, it follows that either $\alpha_{m_1^v,m} = \alpha_{m_0^v,m}$ (if $\varepsilon(m, x^{(<t)}) = 0$) or $\delta \leq \alpha_{m_1^v,m}, \alpha_{m_0^v,m} \leq 1 - \delta$ (if $\varepsilon(m, x^{(<t)}) \neq 0$). Thus, to bound the right hand side in (4.1), we can use the following claim

Claim 4.9

Let P and Q be two distributions over a universe Σ such that for any element $m \in \Sigma$, $\delta \leq P(m), Q(m) \leq 1 - \delta$. Then $\sum_{m\in\Sigma}|P(m) - Q(m)| \leq 2(1 - 2\delta)$.

Proof of Claim 4.9. Let $\Sigma_+ := \{m : P(m) > Q(m)\}$. We may write

$$\begin{aligned}\sum_{m\in\Sigma}|P(m)-Q(m)| &= \sum_{m\in\Sigma_+}(P(m)-Q(m)) + \sum_{m\in setminus\Sigma_+}(Q(m)-P(m))\\ &= P(\Sigma_+)-Q(\Sigma_+)+Q(\Sigma\setminus\Sigma_+)-P(\Sigma\setminus\Sigma_+)\\ &= 2(P(\Sigma_+)-Q(\Sigma_+)),\end{aligned}$$

where in the last line we used the fact that $Q(\Sigma\setminus\Sigma_+)-P(\Sigma\setminus\Sigma_+) = 1-Q(\Sigma_+)-1+P(\Sigma_+) = P(\Sigma_+)-Q(\Sigma_+)$. We now distinguish two cases. **Case** 1.If Σ_+ is a singleton, $\Sigma_+ = \{m^*\}$, then $P(\Sigma_+)-Q(\Sigma_+) = P(m^*)-Q(m^*) \le 1-2\delta$, by assumption. **Case** 2**.** Otherwise, $|\Sigma_+| \ge 2$ and $2\sum_{m\in\Sigma_+}(P(m)-Q(m)) \le 2-2\sum_{m\in\Sigma_+}Q(m) \le 2(1-\delta|\Sigma_+|) \le 2(1-2\delta)$, using the fact that for any m, $Q(m)\ge\delta$, and the fact that P is a probability measure. This completes the proof of Claim 4.9. □

Applying Claim 4.9 for a fixed $v \in S$ to distributions $(\alpha_{m_0^v,m})_m$ and $(\alpha_{m_1^v,m})_m$, we obtain

$$\frac{1}{n}\sum_{m\in\Sigma}\sum_{v\in S}|\alpha_{m_1^v,m}-\alpha_{m_0^v,m}| \le \frac{1}{n}2\sum_{v\in S}(1-2\delta) \le \frac{2(1-2\delta)s}{n}.$$

Hence, we have $\Pi(\mathcal{P}) \in \texttt{ACDT}\left(\frac{2(1-2\delta)\mathtt{s}}{\mathtt{n}},\delta\right)$, establishing Lemma 4.8. □

Thanks to Claims 4.7 and Lemma 4.8, Theorem 4.5 regarding the *broadcast-$\mathcal{PULL}$* model becomes a direct consequence of Theorem 4.6 on the adaptive coin distinguishing task, taking $\varepsilon = \frac{2(1-2\delta)s}{n}$. More precisely, the assumption $\frac{(1-2\delta)}{\delta sn} \le c$ for some small constant c, ensures that $\frac{\varepsilon}{\delta} \le c$ as required by Theorem 4.6. The lower bound $\Omega\left(\frac{\varepsilon^2}{\delta}\right)$ corresponds to $\Omega\left(\frac{n^2\delta}{(1-2\delta)^2s^2}\right)$. This concludes the proof of Theorem 4.5. □

To establish our results it remains to prove Theorem 4.6.

4.3.3 Proof of Theorem 4.6

We start by recalling some facts from Hypothesis Testing. First let us recall two standard notions of (pseudo) distances between probability distributions. Given two discrete distributions P_0, P_1 over a probability space Ω with the same support[6], the *total variation distance* is defined as $TV(P_0,P_1) := \frac{1}{2}\sum_{x\in\Omega}|P_0(x)-P_1(x)|$, and the Kullback-Leibler divergence $KL(P_0,P_1)$ is defined[7] as $KL(P_0,P_1) := \sum_{x\in\Omega}P_0(x)\log\frac{P_1(x)}{P_0(x)}$.

The following lemma shows that, when trying to discriminate between distributions P_0, P_1, the total variation relates to the smallest error probability we can hope for.

[6]The assumption that the support is the same is not necessary but it is sufficient for our purposes, and is thus made for simplicity's sake.

[7]We use the notation $\log(\cdot)$ to denote the base 2 logarithms, i.e., $\log_2(\cdot)$ and for a probability distribution P, use the notation $P(x)$ as a short for $P(X = x)$.

Lemma 4.10: Neyman-Pearson [108, Lemma 5.3 and Proposition 5.4]

Let P_0, P_1 be two distributions. Let X be a random variable of law either P_0 or P_1. Consider a (possibly probabilistic) mapping $f : \Omega \to \{0, 1\}$ that attempts to "guess" whether the observation X was drawn from P_0 (in which case it outputs 0) or from P_1 (in which case it outputs 1). Then, we have the following lower bound,

$$P_0\left(f(X) = 1\right) + P_1\left(f(X) = 0\right) \geq 1 - TV(P_0, P_1). \tag{4.2}$$

The total variation is related to the KL divergence by the following inequality.

Lemma 4.11: Pinsker [108, Lemma 5.8]

For any two distributions P_0, P_1,

$$TV(P_0, P_1) \leq \sqrt{KL\left(P_0, P_1\right)}. \tag{4.3}$$

We are now ready to prove the theorem.

Proof of Theorem 4.6. Let us define $P_\eta\left(\,\cdot\,\right) = P\left(\,\cdot \mid \text{"correct distribution is } \eta\text{"}\right)$ for $\eta \in \{0, 1\}$. We denote $P_\eta^{(\leq t)}$, $\eta \in \{0, 1\}$, the two possible distributions of $X^{(\leq t)}$. We refer to $P_0^{(\leq t)}$ as the distribution of *type* 0 and to $P_1^{(\leq t)}$ as the distribution of *type* 1. Furthermore, we define the *correct type* of a sequence of observations $X^{(\leq t)}$ to be 0 if the observations are sampled from $P_0^{(\leq t)}$, and to be 1 if they are sampled from $P_1^{(\leq t)}$.

After t observations $x^{(\leq t)} = (x^{(1)}, \ldots, x^{(t)})$ we have to decide whether the distribution is of type 0 or 1. Our goal is to maximize the probability of guessing the type of the distribution, observing $X^{(\leq t)}$, which means that we want to minimize

$$f = \sum_{\eta \in \{0,1\}} P_\eta\left(f(X^{(\leq t)}) = 1 - \eta\right) P\left(\text{"correct type is } \eta\text{"}\right). \tag{4.4}$$

Recall that the correct type is either 0 or 1 with probability $\frac{1}{2}$. Thus, the error probability described in (4.4) becomes

$$\frac{1}{2} P_0\left(f(X^{(\leq t)}) = 1\right) + \frac{1}{2} P_1\left(f(X^{(\leq t)}) = 0\right). \tag{4.5}$$

By combining Lemmas 4.10 and 4.11 with $X = X^{(\leq t)}$ and $P_\eta = P_\eta^{(\leq t)}$ for $\eta = 0, 1$, we get the following Theorem. Although for convenience we think of f as a deterministic function, it could in principle be randomized.

Theorem 4.12

Let f be any guess function. Then

$$P_0\left(f(X^{(\leq t)}) = 1\right) + P_1\left(f(X^{(\leq t)}) = 0\right) \geq 1 - \sqrt{KL\left(P_0^{(\leq t)}, P_1^{(\leq t)}\right)}. \qquad (4.6)$$

Theorem 4.12 implies that for the probability of error to be small, it must be the case that the term $KL\left(P_0^{(\leq t)}, P_1^{(\leq t)}\right)$ is large. Our next goal is therefore to show that in order to make this term large, t must be large.

Note that $P_\eta^{(\leq T)}$ for $\eta \in \{0,1\}$ cannot be written as the mere product of the marginal distributions of the $X^{(t)}$s, since the observations at different times may not necessarily be independent. Nevertheless, we can still express the term $KL(P_0^{(\leq T)}, P_1^{(\leq T)})$ as a sum, using the Chain Rule for KL divergence. It yields

$$KL(P_0^{(\leq T)}, P_1^{(\leq T)}) = \sum_{t \leq T} KL(P_0(x^{(t)} \mid x^{(<t)}), P_1(x^{(t)} \mid x^{(<t)})) \qquad (4.7)$$

$$:= \sum_{x^{(<t)} \in \Sigma^{t-1}} P_0(x^{(<t)}) \sum_{x^{(t)} \in \Sigma} P_0(x^{(t)} \mid x^{(<t)}) \log \frac{P_0(x^{(t)} \mid x^{(<t)})}{P_1(x^{(t)} \mid x^{(<t)})}.$$

$$= \sum_{x^{(<t)} \in \Sigma^{t-1}} P_0(x^{(<t)}) \sum_{m \in \Sigma} P_0(X_0^{(t)} = m \mid x^{(<t)}) \log \frac{P(X_0^{(t)} = m \mid x^{(<t)})}{P(X_1^{(t)} = m \mid x^{(<t)})}. \qquad (4.8)$$

Since we are considering an instance of $\texttt{ACDT}(\varepsilon, \delta)$, we have

- $d_\varepsilon(x^{(<t)}) = \sum_{m \in \Sigma} |\varepsilon(m, x^{(<t)})| \leq \varepsilon$, and
- for every $m \in \Sigma$ such that $\varepsilon(m, x^{(<t)}) \neq 0$, it holds that $\delta \leq P_\eta(X_0^{(t)} = m \mid x^{(<t)})$ for $\eta \in \{0,1\}$.

We make use of the previous facts to upper bound the KL divergence terms in the right hand side of (4.8), as follows.

$$KL(P_0(x^{(t)} \mid x^{(<t)}), P_1(x^{(t)} \mid x^{(<t)})) \qquad (4.9)$$

$$= \sum_{x^{(<t)} \in \Sigma^{t-1}} P_0(x^{(<t)}) \sum_{m \in \Sigma} \left(P(X_0^{(t)} = m \mid x^{(<t)}) \log \frac{P(X_0^{(t)} = m \mid x^{(<t)})}{P(X_0^{(t)} = m \mid x^{(<t)}) + \varepsilon(m, x^{(<t)})} \right) \qquad (4.10)$$

$$= - \sum_{x^{(<t)}} P_0(x^{(<t)}) \sum_{m \in \Sigma} \left(P(X_0^{(t)} = m \mid x^{(<t)}) \log \left(1 + \frac{\varepsilon(m, x^{(<t)})}{P(X_0^{(t)} = m \mid x^{(<t)})} \right) \right). \qquad (4.11)$$

Recall that we assume $\frac{\varepsilon(m,x^{(<t)})}{P(X_0^{(t)}=m|x^{(<t)})} \leq \frac{\varepsilon(m,x^{(<t)})}{\delta} \leq \frac{\varepsilon}{\delta}$. We make use of the following claim, which follows from the Taylor expansion of $\log(1+u)$ around 0.

Claim 4.13

Let $x \in [-a,a]$ for some $a \in (0,1)$. Then $|\log(1+x) - x + x^2/2| \leq \frac{x^3}{3(1-a)^3}$.

Using Claim 4.13 with $a = \frac{\varepsilon}{\delta}$, we can bound the inner sum appearing in (4.11) from above and below with

$$\frac{1}{\ln 2}\sum_{m\in\Sigma}\left(\varepsilon(m,x^{(<t)}) - \frac{1}{2}\frac{(\varepsilon(m,x^{(<t)}))^2}{P(X_0^{(t)}=m \mid x^{(<t)})} \pm \frac{\delta^3}{3(\delta-\varepsilon)^3}\left(\frac{(\varepsilon(m,x^{(<t)}))^3}{P(X_0^{(t)}=m \mid x^{(<t)})^2}\right)\right). \quad (4.12)$$

Since $\sum_m |\varepsilon(m,x^{(<t)})| \leq \varepsilon$, we also have that $\sum_m (\varepsilon(m,x^{(<t)}))^2 \leq \varepsilon^2$. The latter bound, together with the fact that $P(X_0^{(t)}=\tilde{m} \mid x^{(<t)}) \geq \delta$ for any $\tilde{m} \in \Sigma$ such that $\varepsilon(\tilde{m},x^{(<t)}) \neq 0$, implies

$$\sum_m \frac{(\varepsilon(m,x^{(<t)}))^2}{P(X_0^{(t)}=m \mid x^{(<t)})} \leq \frac{\varepsilon^2}{\delta}\ . \quad (4.13)$$

Finally, we can similarly bound the term $\sum_{m\in\Sigma}\left((\varepsilon(m,x^{(<t)}))^3/P(X_0^{(t)}=m \mid x^{(<t)})^2\right)$ with

$$\sum_{m\in\Sigma}\left((\varepsilon(m,x^{(<t)}))^3/P(X_0^{(t)}=m \mid x^{(<t)})^2\right) \leq \frac{\varepsilon^3}{\delta^2}. \quad (4.14)$$

Recall that $\sum_m \varepsilon(m,x^{(<t)}) = 0$, thus the first term in (4.12) disappears. Hence, substituting the bounds (4.13) and (4.14) in (4.12), we have

$$\begin{aligned}\left|\log\left(1+\frac{\varepsilon(m,x^{(<t)})}{P(X_0^{(t)}=m \mid x^{(<t)})}\right)\right| &\leq \frac{1}{\ln 2}\left(\frac{1}{2}\frac{\varepsilon^2}{\delta} + \frac{\delta\varepsilon^3}{3(\delta-\varepsilon)^3}\right)\\ &\leq \frac{1}{\ln 2}\left(\frac{1}{2} + \frac{\delta^2\varepsilon}{3(\delta-\varepsilon)^3}\right)\frac{\varepsilon^2}{\delta}. \quad (4.15)\end{aligned}$$

If we define the right hand side (4.15) to be $W(\varepsilon,\delta)$ and we substitute the previous bound in (4.11), we get

$$KL(P_0(x^{(t)} \mid x^{(<t)}), P_1(x^{(t)} \mid x^{(<t)})) \leq W(\varepsilon,\delta), \quad (4.16)$$

and combining the previous bound with (4.7), we can finally conclude that for any integer T, we have $KL(P_0^{(\leq T)}, P_1^{(\leq T)}) \leq T \cdot W(\varepsilon,\delta)$. Thus, from Theorem 4.12 and the latter bound, it follows that the error under a uniform prior of the source type, as defined in (4.5), is at least

$$\begin{aligned}\frac{1}{2}P_0\left(f(X^{(\leq t)})=1\right) + \frac{1}{2}P_1\left(f(X^{(\leq t)})=0\right) &\geq \frac{1}{2} - \frac{1}{2}\sqrt{KL(P_0^{(\leq T)}, P_1^{(\leq T)})} \quad (4.17)\\ &\geq \frac{1}{2} - \frac{1}{2}\sqrt{T\cdot W(\varepsilon,\delta)}. \quad (4.18)\end{aligned}$$

Hence, the number of samples T needs to be greater than $\frac{1}{9}\frac{1}{W(\varepsilon,\delta)} = \frac{\ln 2}{9}\left(\frac{6(\delta-\varepsilon)^3}{\delta^3-\delta^2\varepsilon+3\delta\varepsilon^2-\varepsilon^3}\right)\frac{\delta}{\varepsilon^2}$ to allow the possibility that the error be less than $1/3$.

In particular, if we assume that $10\varepsilon < \delta$, then we can bound $\frac{\delta^2\varepsilon}{3(\delta-\varepsilon)^3} \leq \frac{\delta^3}{10}\cdot\frac{1}{3(9/10)^3\delta^3} \leq \frac{100}{2187}$. It follows that (4.15) can be bounded with $W(\varepsilon,\delta) \leq \frac{1}{\ln 2}\left(\frac{1}{2}+\frac{100}{2187}\right) \leq 0.79$, and so $\frac{1}{9}\frac{1}{W(\varepsilon,\delta)} \geq 0.14\cdot\frac{\delta}{\varepsilon^2} = \Omega\left(\frac{\delta}{\varepsilon^2}\right)$. This completes the proof of Theorem 4.6 and hence of Theorem 4.5. □

Chapter 5

Minimizing Message Size in Stochastic Communication Patterns: Fast Self-Stabilizing Protocols with 3 bits

5.1 Introduction

5.1.1 Background and Motivation

In this Chapter as in the previous one, we study distributed systems composed of limited agents that interact in a stochastic fashion to collectively perform a given task. In Chapter 4, the emphasis was put on the effect of noise. Here we will focus on other types of constraints: namely, self-stabilization, and message size.

As many natural systems appear to be more restricted by their communication abilities than by their memory capacities [66, 2, 58], we seek to understand what tasks can be achieved while revealing as few bits per interaction as possible[1]. In dealing with such an existential question, we do not claim that our solution represents actual plausible strategies employed in nature, yet we believe that such mathematical results can be helpful in understanding the limitations of natural systems.

The notion of *self-stabilization* [46], stemming from distributed computing asks that agents converge to a correct configuration from any initial configuration of states. This notion is

[1]We note that stochastic communication patterns such as $\mathcal{PULL}$ or $\mathcal{PUSH}$ are inherently sensitive to congestion issues. Indeed, in such models it is unclear how to simulate a protocol that uses large messages while using only small size messages. For example, the straightforward strategy of breaking a large message into small pieces and sequentially sending them one after another does not work, since one typically cannot make sure that the small messages reach the same destination. Hence, reducing the message size may have a profound impact on the running time, and perhaps even on the solvability of the problem at hand.

relevant in unstable setting, where the environment is constantly evolving. Self-stabilization can be thought as a way to overcome the fact that there is no clear *start* for the algorithmic process. Such a well defined origin of time gives agents knowledge of their respective initial states.

Self-stabilizing Bit Dissemination. Disseminating information from one or several sources to the rest of the population is one of the most fundamental building blocks in distributed computing [36, 40, 45, 47, 87], and an important primitive in natural systems [113, 106, 109]. Here, we focus on the *Majority Bit Dissemination* problem defined as follows. We consider a population of n agents. The population may contain multiple *source agents* which are specified by a designated bit in the state of an agent indicating whether the agent is a source or not. Each source agent holds a binary *input bit*, however, sources may not necessarily agree on their input bits. In addition, each agent holds a binary *output bit* (also called *opinion*). The goal of all agents is to converge their opinion on the majority bit among the initial input bits of the sources, termed b_{maj}. This problem aims to capture scenarios in which some individuals view themselves as informed, but some of these agents could also be wrong, or not up-to-date. Such situations are common in nature [43, 106] as well as in man-made systems. The number of sources is termed k. We do not assume that agents know the value k, or that sources know whether they are in the majority or minority (in terms of their input bit). For simplicity, to avoid dealing with the case that the fraction of the majority input bit among sources is arbitrarily close to that of the minority input bit, we shall guarantee convergence only when the fraction of source agents holding the majority input bit is bounded away from $1/2$.

The particular case where we are promised to have $k = 1$ is called Bit Dissemination, for short. In this case we have a single source agent that aims to disseminate its input bit b to the rest of the population, and there are no other sources introducing a conflicting opinion. Note that this problem has been studied extensively in different models under different names (e.g., *Broadcast* or *Rumor Spreading*). A classical example of Bit Dissemination considers the synchronous $\mathcal{PUSH}/\mathcal{PULL}$ communication model, where b can be propagated from the source to all other agents in $\mathcal{O}(\log n)$ rounds, by simply letting each uninformed agent copy it whenever it sees an informed agent [87]. The correctness of this protocol heavily relies on the initial information held by agents. Such reliability however may be difficult to achieve in dynamic or unreliable conditions. For example, if the source opinions fluctuates stabilizing to its final value, the source may invoke several consecutive executions of the protocol with contradicting initial opinions. This may in turn "infect" other agents with the wrong opinion $1 - b$. If agents do not share a common time notion, it is unclear how to let infected agents distinguish their current wrong opinion from the more "fresh", correct opinion.

5.1.2 Technical Difficulties and Intuition

Consider the Bit Dissemination problem (where we are guaranteed to have a single source agent). This particular case is already difficult in the self-stabilizing context if we are restricted

to use $O(1)$ bits per interaction. As hinted above, a main difficulty lies in the fact that agents do not necessarily share a common time notion. Indeed, it is easy to see that if all agents share the same clock, then convergence can be achieved in $\mathcal{O}(\log n)$ time with high probability (w.h.p.), i.e, with a probability of at least $1 - n^{-\Omega(1)}$, and using two bits per interaction.

Self-stabilizing Bit Dissemination ($k = 1$) with 2 bits per interaction, assuming synchronized clocks. The source sets her output bit to be her input bit b. In addition to communicate its output bit b_u, each agent u stores and communicates a *certainty* bit c_u. Informally, having a certainty bit equal to 1 indicates that the agent is certain of the correctness of its output bit. The source's certainty bit is always set to 1. Whenever a non-source agent v observes u and sees the tuple (b_u, c_u), where $c_u = 1$, it copies the output and certainty bits of u (*i.e.,* sets $b_v = b_u$ and $c_v = 1$). In addition, all non-source agents count rounds, and reset their certainty bit to 0 simultaneously every $T = \mathcal{O}(\log n)$ rounds. The reset allows to get rid of "old" output bits that may result from applying the protocol before the source's output bit has stabilized. This way, from the first time a reset is applied after the source's output bit has stabilized, the correct source's output bit will propagate to all agents within T rounds, w.h.p. Note however, that if agents do not share a consistent notion of time they cannot reset their certainty bit to zero simultaneously. In such cases, it is unclear how to prevent agents that have just reset their certainty bit to 0 from being "infected" by "misleading" agents, namely, those that have the wrong output bit and certainty bit equal to 1.

Self-stabilizing Bit Dissemination ($k = 1$) with a single bit per interaction, assuming synchronized clocks. Under the assumption that all agents share the same clock, the following trick shows how to obtain convergence in $\mathcal{O}(\log n)$ time and using only a single bit per message, namely, the output bit. As before, the source sets her output bit to be her input bit b. Essentially, agents divide time into phases of some prescribed length $T = \mathcal{O}(\log n)$, each of them being further subdivided into 2 subphases of length $T/2$. In the first subphase of each phase, non-source agents are *sensitive* to opinion 0. This means that whenever they see a 0 in the output bit of another agent, they turn their output bit to 0, but if they see 1 they ignore it. Then, in the second subphase of each phase, they do the opposite, namely they switch their output bit to 1 as soon as they see a 1 (see Figure 5.1). Consider the first phase starting after initialization. If $b = 0$ then within one complete subphase $[1, T/2]$, every output bit is 0 w.h.p., and remains there forever. Otherwise, if $b = 1$, when all agents go over a subphase $[T/2 + 1, T]$ all output bits are set to 1 w.h.p., and remain 1 forever. Note that a common time notion is required to achieve correctness.

The previous protocol indicates that the self-stabilizing Bit Dissemination problem is highly related to the self-stabilizing *Clock Synchronization* problem, where each agent internally stores a clock modulo $T = \mathcal{O}(\log n)$ incremented at every round and, despite having arbitrary initial states, all agents should converge on sharing the same value of the clock. Indeed, given such a protocol, one can obtain a self-stabilizing Bit Dissemination protocol by running the Clock

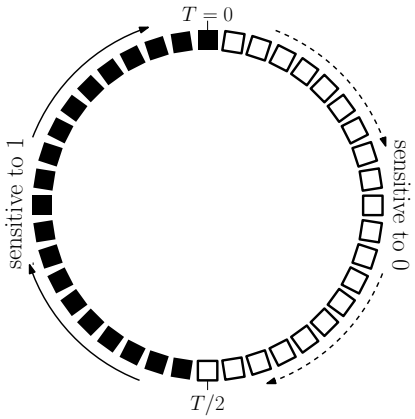

Figure 5.1: The division in subphases used for self-stabilizing Bit Dissemination with a clock. During the first half, between times 1 and $T/2$, agents are sensitive to 0. Then they are sensitive to 1.

Synchronization protocol in parallel to the last example protocol. This parallel execution costs only an additional bit to the message size and a $\mathcal{O}(\log n)$ additive term to the time complexity over the complexities of the Clock Synchronization protocol.

Intuition behind the self-stabilizing Clock Synchronization algorithm. Our technique for obtaining the Clock Synchronization protocol is based on a compact recursive use of the stabilizing consensus protocol proposed by Doerr et al. [48] through our Message Reduction Theorem (Theorem 5.6).

In Section 5.2.2, we describe a simple protocol called Syn-Simple that uses $\mathcal{O}(\log T)$ bits per message. In Syn-Simple, each agent u maintains a clock $C_u \in [0, T-1]$. At each round, each agent u displays the opinion of her clock, pulls 2 other such clock opinions, and updates her clock as the bitwise majority of the two clocks she pulled and her own. Then the clock C_u is incremented. This protocol essentially amounts to running the protocol of Doerr et al. on each bit separately and in parallel, and self-stabilizes in $\mathcal{O}(\log T \log n)$ rounds w.h.p. (Proposition 5.2.1).

We want to apply a strategy similar to Syn-Simple, while using only $\mathcal{O}(1)$ many bits per interaction. The core technical ingredient, made rigorous in the Message Reduction Theorem, is that a certain class of protocols using messages of ℓ bits, to which Syn-Simple belongs, can be emulated by another protocol which uses $\lceil \log \ell \rceil + 1$ bits only. The idea is to build a clock modulo ℓ using Syn-Simple itself on $\lceil \log \ell \rceil$ bits and sequentially display one bit of the original ℓ-bit message according to such clock. Thus, by applying such strategy to Syn-Simple itself, we use a smaller clock modulo $\ell' \ll \ell$ to synchronize a clock modulo ℓ. Iterating such process, in Section 5.4, we obtain a compact protocol which uses only 3 bits.

5.1.3 Organization of the Chapter

We first define the model with more details than in the Introduction 1.5.4, and give our results afterwards in Section 5.1.5. After some necessary preliminaries (Section 5.2), we present the compiler (the Message Reduction Theorem) that allows to reduce the message size of our protocols in Section 5.3. Section 5.4 is devoted to the clock synchronization protocol and Section 5.5 builds on it to derive our protocol for Majority Bit Dissemination.

5.1.4 The Model

The communication model. We adopt the synchronous $\mathcal{PULL}$ model [18, 45]. Specifically, in the $\mathcal{PULL}(\eta)$ model, communication proceeds in discrete rounds. In each round, each agent u "observes" η arbitrary other agents, chosen uniformly at random (with replacement), which we abbreviate as u.a.r., among all agents, including herself. (We often omit the parameter η when it is equal to 2). When an agent u "observes" another agent v, she can peek into a designated *visible part* of v's memory. If several agents observe an agent v at the same round then they all see the same visible part. The *message size* denotes the number of bits stored in the visible part of an agent. We denote with $\mathcal{PULL}(\eta, \ell)$ the $\mathcal{PULL}(\eta)$ model with message size ℓ. We are primarily interested in message size that is independent of n, the number of agents.

Agents. We assume that agents do not have unique identities, that is, the system is *anonymous*. We do not aim to minimize the (non-visible) memory requirement of the agent, yet our constructions can be implemented with relatively short memory, using $\mathcal{O}(\log \log n)$ bits. We assume that each agent internally stores a clock modulo some integer $T = \mathcal{O}(\log n)$, which is incremented at every round.

Majority Bit Dissemination problem. We assume a system of n agents each having an internal state that contains an *indicator bit* which indicates whether or not the agent is a *source*. Each source holds a binary *input bit* and each agent (including sources) holds a binary *opinion*. . Note that having the indicator bit equal to 1 is equivalent to possessing an input bit: both are exclusive properties of source nodes. However, we keep them distinct for a clearer presentation. The number of sources (i.e., agents whose indicator bit is 1) is denoted by k. We denote by k_0 and k_1 the number of sources whose input bit is initially set to 1 and 0, respectively. Assuming $k_1 \neq k_0$, we define the *majority bit*, termed b_{maj}, as 1 if $k_1 > k_0$ and 0 if $k_1 < k_0$. Source agents know that they are sources (using the indicator bit) but they do not know whether they hold the majority bit. The parameters k, k_1 or k_0 are not known to the sources or to any other agent. It is required that the opinions of agents converge to the majority bit b_{maj}.

We note that agents hold their output and indicator bits privately, and we do not require them to necessarily reveal these bits publicly (in their visible parts) unless they wish to. To avoid dealing with the cases where the number of sources holding the majority bit is arbitrarily

close to $\frac{k}{2}$, we shall guarantee correctness (w.h.p.) only if the fraction of sources holding the majority is bounded away from $\frac{1}{2}$, i.e., only if $|\frac{k_1}{k_0} - 1| > \varepsilon$, for some positive constant ε. When $k = 1$, the problem is called *Bit Dissemination*, for short. Note that in this case, the single source agent holds the bit b_{maj} to be disseminated and there is no other source agent introducing a conflicting opinion.

T-Clock Synchronization. Let T be an integer. In the *T-Clock Synchronization* problem, each agent maintains a *clock* modulo T that is incremented at each round. The goal of agents is to converge on having the same value in their clocks modulo T. (We may omit the parameter T when it is clear from the context.)

Probabilistic self-stabilization and convergence. Self-stabilizing protocols are meant to guarantee that the system eventually converges to a *legal* configuration regardless of the initial states of the agents [46]. Here we use a slightly weaker notion, called *probabilistic self-stabilization*, where stability is guaranteed w.h.p. [16]. More formally, for the Clock Synchronization problem, we assume that *all* states are initially set by an adversary. For the Majority Bit Dissemination problem, we assume that *all* states are initially set by an adversary except that it is assumed that the agents know their total number n, and that this information is not corrupted.

In the context of T-Clock Synchronization, a legal configuration is reached when all clocks show the same time modulo T, and in the Majority Bit Dissemination problem, a legal configuration is reached when all agents output the majority bit b_{maj}. Note that in the context of the Majority Bit Dissemination problem, the legality criterion depends on the initial configuration (that may be set by an adversary). That is, the agents must converge their opinion on the majority of input bits of sources, as evident in the initial configuration.

The system is said to *stabilize* in t rounds if, from any initial configuration, with high probability, within t rounds it reaches a legal configuration and remains legal for at least some polynomial time [16, 18, 48]. In fact, for the self-stabilizing Bit Dissemination problem, if there are no conflicting source agents holding a minority opinion (such as in the case of a single source agent), then our protocols guarantee that once a legal configuration is reached, it remains legal indefinitely. Note that, for any of the problems, we do not require that each agent irrevocably commits to a final opinion but that eventually agents arrive at a legal configuration without necessarily being aware of that.

5.1.5 Our Results

Our main results are the following.

Theorem 5.1

Fix an arbitrarily small constant $\varepsilon > 0$. There exists a protocol, called Syn-Phase-Spread, which solves the Majority Bit Dissemination problem in a self-stabilizing manner in $\tilde{\mathcal{O}}(\log n)$ rounds[a] w.h.p using 3-bit messages, provided that the majority bit is supported by at least a fraction $\frac{1}{2} + \varepsilon$ of the source agents.

[a] With a slight abuse of notation, with $\tilde{\mathcal{O}}(f(n)g(T))$ we refer to $f(n)g(T)\log^{\mathcal{O}(1)}(f(n))\log^{\mathcal{O}(1)}(g(T))$. All logarithms are in base 2.

Theorem 5.1 is proved in Section 5.5. The core ingredient of Syn-Phase-Spread is our construction of an efficient self-stabilizing T-Clock Synchronization protocol, which is used as a black-box. For this purpose, the case that interests us is when $T = \tilde{\mathcal{O}}(\log n)$. Note that in this case, the following theorem, proved in Section 5.4, states that the convergence time of the Clock Synchronization algorithm is $\tilde{\mathcal{O}}(\log n)$.

Theorem 5.2

Let T be an integer. There exists a self-stabilizing T-Clock Synchronization protocol, called Syn-Clock, which employs only 3-bit messages, and synchronizes clocks modulo T within $\tilde{\mathcal{O}}(\log n \log T)$ rounds w.h.p.

In addition to the self-stabilizing context our protocols can tolerate the presence of Byzantine agents, as long as their number is[2] $\mathcal{O}(n^{1/2-\varepsilon})$. However, in order to focus on the self-stabilizing aspect of our results, in this work we do not explicitly address the presence of Byzantine agents.

The proofs of both Theorem 5.2 and Theorem 5.1 rely on recursively applying a new general compiler which can essentially transform any self-stabilizing algorithm with a certain property (called "the bitwise-independence property") that uses ℓ-bit messages to one that uses only $\lceil \log \ell \rceil + 1$-bit messages, while paying only a small penalty in the running time. This compiler is described in Section 5.3, in Theorem 5.6, which is also referred as "the Message Reduction Theorem". The structure between our different lemmas and results is summarized in the picture below, Figure 5.2.

It remains an open problem, both for the self-stabilizing Bit Dissemination problem and for the self-stabilizing Clock Synchronization problem, whether the message size can be reduced to 2 bits or even to 1 bit, while keeping the running time poly-logarithmic.

[2] Specifically, it is possible to show that, as a corollary of our analysis and the fault-tolerance property of the analysis in [48], if $T \leq poly(n)$ then Syn-Clock can tolerate the presence of up to $\mathcal{O}(n^{1/2-\varepsilon})$ Byzantine agents for any $\varepsilon > 0$. In addition, Syn-Phase-Spread can tolerate $\min\{(1-\varepsilon)(k_{maj}-k_{min}), n^{1/2-\varepsilon}\}$ Byzantine agents, where k_{maj} and k_{min} are the number of sources supporting the majority and minority opinions, respectively. Note that for the case of a single source ($k = 1$), no Byzantine agents are allowed; indeed, a single Byzantine agent pretending to be the source with the opposite opinion can clearly ruin any protocol.

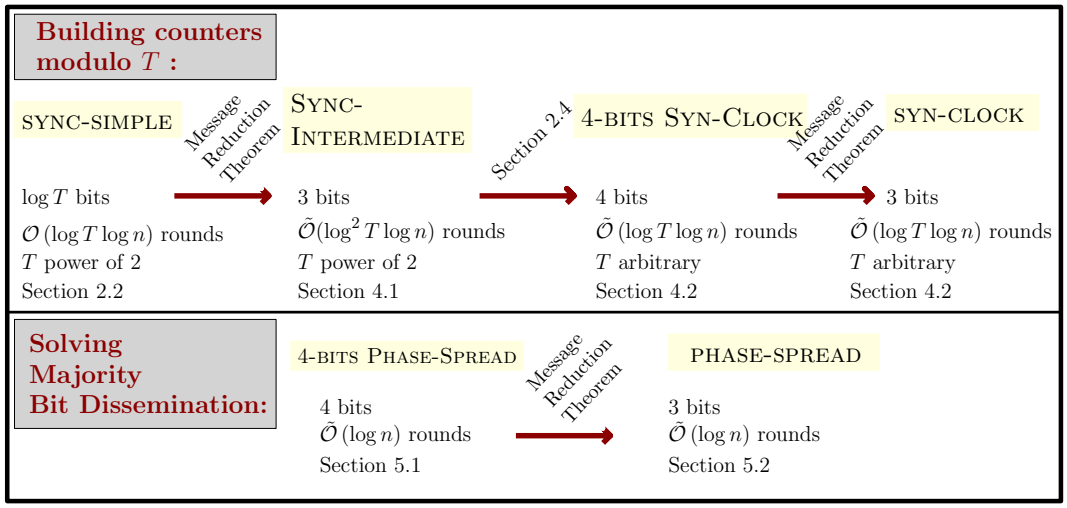

Figure 5.2: The structure of our arguments. Note that the Message Reduction Theorem is used on three occasions.

5.1.6 Related Work

The computational study of abstract systems composed of simple individuals that interact using highly restricted and stochastic interactions has recently been gaining considerable attention in the community of theoretical computer science. Popular models include *population protocols* [7, 12, 9, 15], which typically consider constant size individuals that interact in pairs (using constant size messages) in random communication patterns, and the *beeping* model [2, 58], which assumes a fixed network with extremely restricted communication. Our model also falls in this framework as we consider the $\mathcal{PULL}$ model [45, 87, 89] with constant size messages. So far, despite interesting works that consider different fault-tolerant contexts [8, 9, 15], most of the progress in this framework considered non-faulty scenarios.

Information dissemination is one of the most well-studied topics in the community of distributed computing, see, *e.g.,* [8, 36, 45, 47, 48, 63, 87]. Classical examples include the *Broadcast* (also referred to in the literature as *Rumor Spreading*) problem, in which a piece of information residing at one source agent is to be disseminated to the rest of the population, and *majority-consensus* (here, called Majority Bit Dissemination) problems in which processors are required to agree on a common output value which is the majority initial input value among all agents [8, 91] or among a set of designated source agents [63]. An extensive amount of research has been dedicated to study such problems in $\mathcal{PUSH}/\mathcal{PULL}$ based protocols (including the *phone call* model), due to the inherent simplicity and fault-tolerant resilience of such meeting patterns. Indeed, the robustness of $\mathcal{PUSH}/\mathcal{PULL}$ based protocols to weak types of faults, such as crashes of messages and/or agents, or to the presence of relatively few Byzantine agents, has

been known for quite a while [56, 87]. Recently, it has been shown that under the $\mathcal{PUSH}$ model, there exist efficient Broadcast and Majority Bit Dissemination protocols that use a single bit per message and can overcome flips in messages (noise) [63]. The protocols therein, however, heavily rely on the assumption that agents know when the protocol has started. Observe that in a self-stabilizing context, in which the adversary can corrupt the initial clocks setting them to arbitrary times, such an assumption would be difficult to remove while preserving the small message size.

In general, there are only few known self-stabilizing protocols that operate efficiently under stochastic and capacity restricted interactions. An example, which is also of high relevance to this work, is the work of Doerr et al. on *Stabilizing Consensus* [48] operating in the $\mathcal{PULL}$ model. In that work, each agent initially has a state taken out of a set of m opinions and the goal is to converge on one of the proposed states. The proposed algorithm which runs in logarithmic time is based on sampling the states of 2 agents and updating the agent's state to be the median of the 2 sampled states and the current state of the agent (3 opinions in total). Since the total number of possible states is m, the number of bits that must be revealed in each interaction is $\Omega(\log m)$. Another example is the plurality consensus protocol in [18], in which each agent has initially an opinion and we want the system to converge to the most frequent one in the initial configuration of the system. In fact, the Majority Bit Dissemination problem can be viewed as a generalization of the *majority-consensus* problem (i.e. the plurality consensus problem with two opinions), to the case in which multiple agents may initially be unopinionated. In the previous sense, we also contribute to the line of research on the majority-consensus problem [17, 42, 55].

Another fundamental building block is Clock Synchronization [13, 93, 95, 96]. We consider a synchronous system in which clocks tick at the same pace but may not share the same opinion. This version has earlier been studied in e.g., [20, 49, 50, 51, 65, 83] under different names, including "digital Clock Synchronization" and "synchronization of phase-clocks"; We simply use the term "Clock Synchronization". There is by now a substantial line of work on Clock Synchronization problems in a self-stabilizing context [84, 51, 98, 97]. We note that in these papers the main focus is on the resilience to Byzantine agents. The number of rounds and message lengths are also minimized, but typically as a function of the number of Byzantine processors. Our focus is instead on minimizing the time and message complexities as much as possible. The authors in [98, 97] consider mostly a deterministic setting. The communication model is very different than ours, as every agent gets one message from every other agent on each round. Moreover, agents are assumed to have unique identifiers. In contrast, we work in a more restricted, yet randomized communication setting. In [84, 98] randomized protocols are also investigated. We remark that the first protocol we discuss Syn-Simple (Proposition 5.2.1), which relies on a known simple connection between consensus and counting [84], already improves exponentially on the randomized algorithms from [84, 98] in terms of number of rounds, number of memory states, message length and total amount of communication, in the restricted regime where the resilience parameter f satisfies $\log n \leq f \leq \sqrt{n}$. We further note that the works [97, 98] also use a recursive construction for their clocks (although very different

from the one we use in the proof of Theorem 5.2). The induction in [98] is on the resilience parameter f, the number of agents and the clock length together. This idea is improved in [97] to achieve optimality in terms of resilience to Byzantine agents.

To the best of our knowledge there are no previous works on self-stabilizing Clock Synchronization or Majority Bit Dissemination that aim to minimize the message size beyond logarithmic in the $\mathcal{PULL}$ model.

5.2 Preliminaries

5.2.1 A majority Based, Self-Stabilizing Protocol for Consensus on One Bit

Let us recall[3] the stabilizing consensus protocol by Doerr et al. in [48]. In this protocol, called MAJ-CONSENSUS, each agent holds an opinion. In each round each agent looks at the opinions of two other random agents and updates her opinion taking the majority among the bits of the observed agents and its own. Note that this protocol uses only a single bit per interaction, namely, the opinion. The usefulness of MAJ-CONSENSUS comes from its extremely fast and fault-tolerant convergence toward an agreement among agents, as given by the following result [4].

Theorem 5.3: Doerr et al. [48]

From any initial configuration, MAJ-CONSENSUS converges to a state in which all agents agree on the same output bit in $\mathcal{O}(\log n)$ rounds, w.h.p. Moreover, if there are at most $\kappa \leq n^{1/2-\varepsilon}$ Byzantine agents, for any constant $\varepsilon > 0$, then after $\mathcal{O}(\log n)$ rounds all non-Byzantine agents have converged and consensus is maintained for $n^{\Omega(1)}$ rounds w.h.p.

[3] Our protocols will use this protocol as a *black box*. However, we note that the constructions we outline are in fact independent of the choice of consensus protocol, and this protocol could be replaced by other protocols that achieve similar guarantees.

[4] The original statement of [48] says that if at most $\kappa \leq \sqrt{n}$ agents can be corrupted at any round, then convergence happens for all but at most $\mathcal{O}(\kappa)$ agents. Let us explain how this implies the statement we gave, namely that we can replace $\mathcal{O}(\kappa)$ by κ, if $\kappa \leq n^{\frac{1}{2}-\varepsilon}$. Assume that we are in the regime $\kappa \leq n^{\frac{1}{2}-\varepsilon}$. It follows from [48] that all but a set of $\mathcal{O}(\kappa)$ agents reach consensus after $\mathcal{O}(\log n)$ round. This set of size $\mathcal{O}(\kappa)$ contains both Byzantine and non Byzantine agents. However, if the number of agents holding the minority opinion is $\mathcal{O}(\kappa) = \mathcal{O}(n^{1/2-\varepsilon})$, then the expected number of non Byzantine agents that disagree with the majority *at the next round* is in expectation $\mathcal{O}(\kappa^2/n) = \mathcal{O}(n^{-2\varepsilon})$. Thus, by Markov's inequality, this implies, that at the next round consensus is reached among *all non-Byzantine agents* w.h.p. Note also that, for the same reasons, the Byzantine agents do not affect any other non-Byzantine agent for n^{ε} rounds w.h.p.

5.2.2 Protocol Syn-Simple: A simple Protocol with Many Bits per Interaction

We now present a simple self-stabilizing T-Clock Synchronization protocol, called Syn-Simple, that uses relatively many bits per message, and relies on the assumption that T is a power of 2. The protocol is based on iteratively applying a self-stabilizing consensus protocol on each bit of the clock separately, and in parallel.

Formally, each agent u maintains a clock $C_u \in [0, T-1]$. At each round, u displays the opinion of her clock C_u, pulls 2 uniform other such clock opinions, and updates her clock as the bitwise majority of the two clocks it pulled, and her own. Subsequently, the clock C_u is incremented. We present the pseudo code of Syn-Simple in Algorithm 1.

Syn-Simple protocol

1 u samples two agents u_1 and u_2.

2 u updates its clock with the bitwise majority of its clock and those of the sample nodes.

3 u increments its clock by one unit.

Algorithm 1: One round of Syn-Simple, executed by each agent u.

We prove the correctness of Syn-Simple in the next proposition.

Proposition 5.2.1. *Let T be a power of 2. The protocol* Syn-Simple *is a self-stabilizing protocol that uses $\mathcal{O}(\log T)$ bits per interaction and synchronizes clocks modulo T in $\mathcal{O}(\log T \log n)$ rounds w.h.p.*

Proof. Let us look at the least significant bit. One round of Syn-Simple is equivalent to one round of maj-consensus with an extra flipping of the opinion due to the increment of the clock. The crucial point is that all agents jointly flip their bit on every round. Because the function agents apply, maj, is symmetric, it commutes with the flipping operation. More formally, let $\vec{b_t}$ be the vector of the first bits of the clocks of the agents at round t under an execution of Syn-Simple. E.g. $(\vec{b_t})_u$ is the value of the less significant bit of node u's clock at time t. Similarly, we denote by $\vec{c_t}$ the first bits of the clocks of the agents at round t obtained by running a modified version of Syn-Simple in which *time is not incremented* (i.e. we skip line 3 in Algorithm 1). We couple $\vec{b}$ and $\vec{c}$ trivially, by running the two versions on the same interaction pattern (in other words, each agent starts with the same memory and pulls the same agents at each round in both executions). Then, $\vec{b_t}$ is equal to $\vec{c_t}$ when t is even, while is equal to $\vec{b_t} = \mathbf{1} - \vec{c_t}$ when t is odd. Moreover, we know from Theorem 5.3 that $\vec{c_t}$ converge to a stable opinion in a self-stabilizing manner. It follows that, from any initial configuration of states (i.e. clocks), w.h.p, after $\mathcal{O}(\log n)$ rounds of executing Syn-Simple, all agents share the same opinion for their first bit, and jointly flip it in each round. Once agents agree on the

first bit, since T is a power of 2, the increment of time makes them flip the second bit *jointly* once every 2 rounds. More generally, assuming agents agree on the first ℓ bits of their clocks, they *jointly* flip the $\ell+1$'st bit once every 2^ℓ rounds, on top of doing the MAJ-CONSENSUS protocol on that bit. Hence, the same coupling argument shows that the flipping doesn't affect the convergence on bit $\ell+1$. Therefore, $\mathcal{O}(\log n)$ rounds after the first ℓ bits are synchronized, w.h.p. the $\ell+1$'st bit is synchronized as well. The result thus follows by induction. □

5.2.3 The bitwise-independence Property

Our general transformer described in Section 5.3 is useful for reducing the message size of protocols with a certain property called bitwise-independence. Before defining the property we need to define a variant of the $\mathcal{PULL}$ model, which we refer to as the $\mathcal{BIT}$ model. The reason we introduce such a variant is mainly technical, as it appears naturally in our proofs. Thus, unless explicitly stated, we always refer to the $\mathcal{PULL}$ model.

Recall that in the $\mathcal{PULL}(\eta,\ell)$ model, at any given round, each agent u is reading an ℓ-bit message m_{v_j} for each of the η observed agents v_j chosen u.a.r. (in our case $\eta=2$), and then, in turn, u updates her state according to the instructions of a protocol P. Informally, in the $\mathcal{BIT}$ model, each agent u also receives η messages, however, in contrast to the $\mathcal{PULL}$ model where each such message corresponds to one observed agent, in the $\mathcal{BIT}$ model, the i'th bit of each such message is received independently from a (typically new) agent, chosen u.a.r. from all agents.

Definition 5.2.2 (The $\mathcal{BIT}$ model)**.** *In the $\mathcal{BIT}$ model, at each round, each agent u picks $\eta\ell$ agents u.a.r., namely, $v_1^{(1)}, v_2^{(1)}, \ldots v_\ell^{(1)}, \ldots, v_1^{(\eta)}, v_2^{(\eta)}, \ldots v_\ell^{(\eta)}$, and reads $\hat{s_i}^{(j)} = s_i(v_i^{(j)})$, the i-th bit of the visible part of agent $v_i^{(j)}$, for every $i \le \ell$ and $j \le \eta$. For each $j \le \eta$, let $\hat{m}_j(u)$ be the ℓ-bit string $\hat{m}_j(u) := (\hat{s_1}^{(j)}, \hat{s_2}^{(j)}, \ldots, \hat{s_\ell}^{(j)})$. By a slight abuse of language we call the strings $\{\hat{m}_j(u)\}_{j\le\eta}$ the* messages *received by u in the $\mathcal{BIT}$ model.*

Definition 5.2.3 (The *bitwise − independence* property)**.** *Consider a protocol* P *designed to work in the $\mathcal{PULL}$ model. We say that* P *has the bitwise-independence property if its correctness and running time guarantees remain the same under the $\mathcal{BIT}$ model (assuming that given the messages $\{\hat{m}_j(u)\}_{j\le\eta}$ it receives at any round, each agent u performs the same actions that it would have, had it received these messages in the $\mathcal{PULL}$ model).*

Let us first state a fact about protocols having the bitwise-independence property.

Lemma 5.4

Assume we are given two protocols SYN-GENERIC and P, designed to work in the $\mathcal{PULL}$ model, such that

- Protocol SYN-GENERIC synchronizes clocks modulo T for some T and
- Protocol P works assuming agents share a clock modulo T.

Denote by SYN-P the parallel execution of SYN-GENERIC and P, with P using the clock synchronized by SYN-GENERIC. Then

1. If SYN-GENERIC and P are self-stabilizing then so is SYN-P, and the convergence time of SYN-P is at most the sum of convergence times of SYN-GENERIC and P.
2. Finally, if SYN-GENERIC and P have the bitwise-independence property, and P is also self-stabilizing, SYN-P has the bitwise-independence property too.

Proof. The self-stabilizing property of SYN-P and its convergence time directly follow from those of SYN-GENERIC and P (part 1 of the statement). We just need to check the correctness of SYN-P, when run in the $\mathcal{BIT}$ model (part 2 of the statement). The fact that SYN-GENERIC and P are run in parallel means that the part of the message and computations regarding SYN-GENERIC are not affected by those regarding P. This still holds when running the protocol in the $\mathcal{BIT}$ model. Since, by hypothesis, SYN-GENERIC has the independence property, there exists a time τ after which all agents share a synchronized clock modulo T, even in the $\mathcal{BIT}$ model. Thus, after time τ, we can disregard the part of the message corresponding to SYN-GENERIC, and view the execution of SYN-P as simply P. The assumption that P is self-stabilizing and has the independence property implies that, regardless of the nodes' memory states concerning the execution of P at time τ, SYN-P still works in the $\mathcal{BIT}$ model as in the original $\mathcal{PULL}$ model, thus inheriting the bitwise-independence property from SYN-GENERIC and P. □

We next show that the protocol SYN-SIMPLE has the aforementioned bitwise-independence property.

Lemma 5.5

SYN-SIMPLE has the bitwise-independence property.

Proof. Let us start by commenting on SYN-SIMPLE, when run in the usual $\mathcal{PULL}$ model. Let ℓ' be the size of the clocks. Assume the first $i < \ell'$ bits of the clocks have been synchronized. At this stage, the $(i+1)$-st bit of each agent u is flipped every 2^i rounds (from 0 to 1 or from 1 to 0) and updated as the majority of the $(i+1)$-st bit of $C(u)$ and the 2 pulled messages on each round. Since the first i bits are synchronized, the previous flipping is performed by

all agents at the same round. Let us now consider the protocol over the $\mathcal{BIT}$ model. Observe that, in order for SYN-SIMPLE to work, we do not need the bit at index $(i+1)$ to come from the same agent as the bits corresponding to indices $\leq i$, as long as convergence on the first i bits has been achieved. Hence, as is, the reasoning above for the $\mathcal{PULL}$ model holds in the $\mathcal{BIT}$ model as well. □

5.3 A General Compiler that Reduces Message Size

In this section we present a general compiler that allows to implement a protocol P using ℓ-bit messages while using messages of order $\log \ell$ instead, as long as P enjoys the bitwise-independence property. The compiler is based on replacing a message by an index to a given bit of the message. This tool will repeatedly be used in the following sections to obtain our Clock Synchronization and Majority Bit Dissemination algorithms that use 3-bit messages.

Theorem 5.6: the Message Reduction Theorem

Any self-stabilizing protocol P in the $\mathcal{PULL}(\eta, \ell)$ model having the bitwise-independence property, and whose running time is L_{P}, can be emulated by a protocol EMUL(P) which runs in the $\mathcal{PULL}(2, \lceil \log(\frac{\eta}{2}\ell) \rceil + 1)$ model, has running time $\mathcal{O}\left(\log(\eta\ell)\log n\right) + \frac{\eta}{2}\ell L_{\mathrm{P}}$ and has itself the bitwise-independence property.

Remark 5.3.1. *The only reason for designing* EMUL(P) *to run in the* $\mathcal{PULL}(2, \lceil \log(\frac{\eta}{2}\ell) \rceil + 1)$ *model in the Message Reduction Theorem is the consensus protocol we adopt,* MAJ-CONSENSUS*, which works in the* $\mathcal{PULL}(2)$ *model.*

In fact, EMUL(P) *can be adapted to run in the* $\mathcal{PULL}(1, \lceil \log(\eta\ell) \rceil + 1)$ *model by using a consensus protocol working in the* $\mathcal{PULL}(1)$ *model. However, no self-stabilizing binary consensus protocol in the* $\mathcal{PULL}(1)$ *model with the same performances as* MAJ-CONSENSUS *is currently known.*

Proof of Theorem 5.6. Let $s(u) \in \{0,1\}^\ell$ be the message displayed by an agent u under P at a given round. For simplicity's sake, in the following we assume that η is even, the other case is handled similarly. In EMUL(P), agent u keeps the message $s(u)$ privately, and instead displays a clock $C(u)$ written on $\lceil \log(\frac{\eta}{2}\ell) \rceil$ bits, and one bit of the message $s(u)$, which we refer to as the P-bit. Thus, the total number of bits displayed by the agent operating in EMUL(P) is $\lceil \log(\frac{\eta}{2}\ell) \rceil + 1$. The purpose of the clock $C(u)$ is to indicate to agent u which bit of $s(u)$ to display. In particular, if the counter has value 0, then the 0-th bit (i.e the least significant bit) of $s(u)$ is shown as the P-bit, and so on. In what follows, we refer to $s(u)$ as the *private message* of u, to emphasize the fact that this message is not visible in EMUL(P). See Figure 5.3 for an illustration.

Each round of P executed in the $\mathcal{PULL}(\eta, \ell)$ model by an agent u is emulated by $\frac{\eta}{2}\ell$ rounds of EMUL(P) in the $\mathcal{PULL}(2, \lceil \log(\frac{\eta}{2}\ell) \rceil + 1)$ model. We refer to such $\frac{\eta}{2}\ell$ rounds as a *phase*,

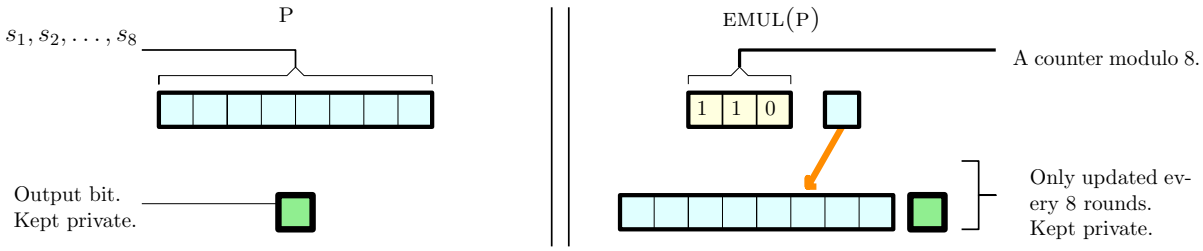

Figure 5.3: On the left is a protocol P using $\ell = 8$ bits in total and pulling only one node per round ($\eta = 1$). On the right is the emulated version EMUL(P) which uses 4 bits only. The bits depicted on the bottom of each panel are kept privately, while the bits on the top are public, that is, appear in the visible part.

which is further divided to $\frac{\eta}{2}$ subphases of length ℓ. Note that since each agent samples 2 agents in a round, the total number of agents sampled by an agent during a phases is $\eta\ell$.

For a generic agent u, a phase starts when its clock $C(u)$ is zero, and ends after a full loop of its clock (i.e. when $C(u)$ returns to zero). Each agent u is running protocol SYN-SIMPLE on the $\lceil \log(\frac{\eta}{2}\ell) \rceil$ bits which correspond to her clock $C(u)$. Note that the phases executed by different agents may initially be unsynchronized, but, thanks to Proposition 5.2.1, the clocks $C(u)$ eventually converge to the same value, for each agent u, and hence all agents eventually agree on when each phase (and subphase) starts.

Let u be an arbitrary agent. Denote by $\hat{s}_1^{(1)}, \hat{s}_2^{(1)}, \ldots \hat{s}_\ell^{(1)}$, ..., $\hat{s}_1^{(\eta)}, \hat{s}_2^{(\eta)}, \ldots \hat{s}_\ell^{(\eta)}$ the P-bits collected by u from agents chosen u.a.r during a phase. Consider a phase and a round $z \in \{1, \cdots, \frac{\eta}{2}\ell\}$ in that phase. Let i and j be such that $z = j \cdot \ell + i$. We view z as round i of subphase $j + 1$ of the phase. On this round, agent u pulls two messages from agents v and w, chosen u.a.r. Once the clocks (and thus phases and subphases) have synchronized, agents v and w are guaranteed to be displaying the ith index of their private messages, namely, the values $s_i(v)$ and $s_i(w)$, respectively. Agent u then sets $\hat{s}_i^{(2j-1)}$ equal to $s_i(v)$ and $\hat{s}_i^{(2j)}$ equal to $s_i(w)$.

In EMUL(P), the messages displayed by agents are only updated after a full loop of C. It therefore follows from the previous paragraph that the P-bits collected by agent u after a full-phase are distributed like the bits collected during one round of P in the $\mathcal{BIT}$ model (see Definition 5.2.2), assuming the clocks are synchronized already.

Correctness. The bitwise-independence property of SYN-SIMPLE (Lemma 5.5), implies that SYN-SIMPLE still works when messages are constructed from the P-bits collected by EMUL(P). Therefore, from Proposition 5.2.1, eventually all the clocks C are synchronized. Since private messages s are only updated after a full loop of C, once the clocks C are synchronized a phase of EMUL(P) corresponds to *one* round of P, executed in the $\mathcal{BIT}$ model. Hence, the hypothesis that P operates correctly in a self-stabilizing way in the $\mathcal{BIT}$ model implies the correctness of EMUL(P).

Running time. Once the clocks $C(u)$ are synchronized, for all agents u, using the first $\lceil \log(\frac{\eta}{2}\ell) \rceil$ bits of the messages, the agents reproduce an execution of P with a multiplicative time-overhead of $\frac{\eta}{2}\ell$. Moreover, from Proposition 5.2.1, synchronizing the clocks $C(u)$ takes $\mathcal{O}(\log(\eta m)\log n)$ rounds. Thus, the time to synchronize the clocks costs only an additive factor of $\mathcal{O}(\log(\eta m)\log n)$ rounds, and the total running time is $\mathcal{O}(\log(\eta m)\log n) + \frac{\eta}{2}\ell \cdot L_{\mathrm{P}}$.

Bitwise-independence property. Protocol Emul(P) inherits the bitwise-independence property from that of Syn-Simple (Lemma 5.5) and P (which has the property by hypothesis): We can apply Lemma 5.4 where Syn-Generic is Syn-Simple and P is the subroutine described above, which displays at each round the bit of P whose index is given by a synchronized clock C modulo ℓ (i.e. the one produced by Syn-Simple). Observe that the aforementioned subroutine is self-stabilizing, since it emulates P once clocks are synchronized. Then, in the notation of Lemma 5.4, Emul(P) is Syn-P. □

5.4 Self-Stabilizing Clock Synchronization

In Section 5.2.2 we described Syn-Simple - a simple self-stabilizing Clock Synchronization protocol that uses $\log T$ bits per interaction. In this section we describe our main self-stabilizing Clock Synchronization protocol, Syn-3Bits, that uses only 3 bits per interaction. We first assume T is a power of 2. We show how to get rid of this assumption at the end of this section.

Clock Synchronization with 3-bit Messages, Assuming T is a Power of Two. In this section, we show the following result.

Lemma 5.7

Let T be a power of 2. There exists a synchronization protocol Syn-Intermediate which synchronizes clocks modulo T in time $\tilde{\mathcal{O}}\left(\log^2 T \log n\right)$ using only 3-bit messages. Moreover, Syn-Intermediate has the bitwise-independence property.

Before presenting the proof of Lemma 5.7, we need a remark about clocks.

Remark 5.4.1. *In order to synchronize a clock C modulo T, throughout the analysis we often obtain a clock C' modulo T* which is incremented every ℓ rounds. *However, C' can still be translated back to a clock modulo T which is incremented every round, by keeping a third clock C'' modulo ℓ and setting*

$$C = \ell C' + C'' \mod T.$$

Proof of Lemma 5.7. At a high level, we simply apply iteratively the Message Reduction Theorem in order to reduce the message to 3 bits, starting with P = Syn-Simple. A pictorial representation of our recursive protocol is given in Figure 5.4, and a pseudocode is given in Algorithm 2. The pseudocode deviates slightly from the presentation done in the proof, as it makes no use of recursion.

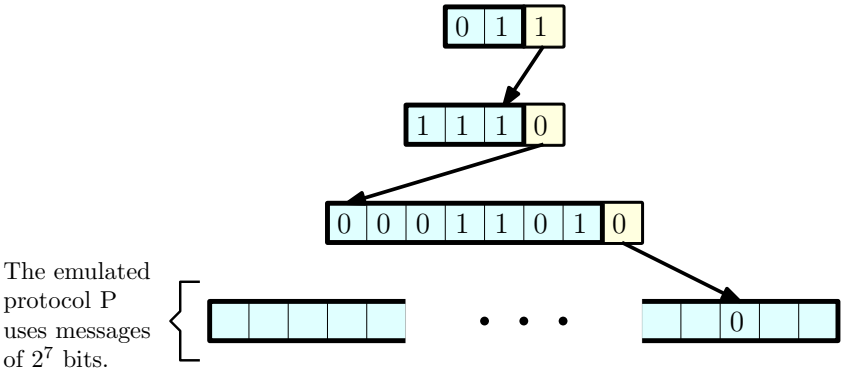

Figure 5.4: A more explicit view of our 3-bit emulation of protocol P, obtained by iterating Lemma 5.6. The down-most layer represents the 2^7-bits message displayed by protocol P. Each layer on the picture may be seen as the message of a protocol emulating P with fewer bits, that is, as we go up on the figure we obtain more and more economical protocols in terms of message length. In particular, the top layer represents the 3-bit message in the final emulation. The left-most part of each message (colored in light blue) encodes a clock. The right-most bit (colored in light yellow) of each message (except the bottom-most one) corresponds to a particular bit of the layer *below* it. The index of this particular displayed bit is given by the value of the clock. Each clock on an intermediate layer is updated only when the clock on the layer *above* completes a loop (i.e., has value 0). The clock on the top-most layer is updated on every round.

Let us consider what we obtain after applying the Message Reduction Theorem the first time to P =Syn-Simple for clocks modulo T. Recall that we assume that T is a power of 2. From Proposition 5.2.1 we know that in this case, the convergence time of Syn-Simple is $L_{\mathrm{P}} = \mathcal{O}(\log T \log n)$, the number of pulled agents at each round is 2 and the number of bits of each message is $\ell = \log T$.

With the emulation produced by the Message Reduction Theorem, the clock used in P =Syn-Simple is incremented only every $\ell = \log T$ rounds. Another way to interpret this is that we obtain a clock modulo $T \cdot \ell$ and using Remark 5.4.1 we can turn it into a counter modulo T that is incremented at each round. Hence, by the running time analysis of the Message Reduction Theorem, we obtain a protocol Emul(P) which synchronizes a clock modulo T in $\mathcal{O}(\log n \log\log T) + \mathcal{O}\left(\log^2 T \log n\right) = \mathcal{O}\left(\log^2 T \log n\right)$ rounds. The message size is reduced from $\log T$ to $\lceil \log\log T \rceil + 1 = \mathcal{O}(\log\log T)$.

By repeatedly applying the Message Reduction Theorem, we reduce the size of the message ℓ as long as $\ell > \lceil \log \ell \rceil + 1$, i.e. as long as $\ell > 3$. The number of repeated application of the Message Reduction Theorem until the message size is 3 is thus of order $\log^* T$.

Let us analyze the running time. Let $\ell_1 = \log T$, $\ell_{i+1} = \lceil \log \ell_i \rceil + 1$ and let $\tau(T) = \tau$ be the smallest integer such that $\ell_\tau = 3$. We apply the Message Reduction Theorem $i \leq \tau$ times,

Syn-Intermediate protocol

MEMORY: Each agent u keeps a sequence of clocks $C_1, \ldots, C_\tau$ and a sequence of bits $b_1, \ldots, b_\tau$. The clock C_1 runs modulo T, the clock C_τ runs modulo 4, and the i-th clock C_i runs modulo $2^{\ell_i - 1}$ (see proof of Lemma 5.7). Each agent u also maintains a sequence of heaps (or some ordered structure) S_i^δ, for each $\delta \in \{1, 2\}$ and $i = 1, \ldots, \tau$.

MESSAGE: u displays C_τ (2 bits) and b_τ (1 bit). For all $i \in [\tau], b_i(u)$ is the $C_i(u)$-th bit of the string obtained concatenating the binary representation of $C_{i-1}(u)$ and $b_{i-1}(u)$.

1 u samples two agents u_1 and u_2.

2 u updates its clock with the bitwise majority of its clock and those of the sampled nodes.

3 u increments its clock by one unit.

4 u sets i^* equal to the maximal $i < \tau$ such that $C_{i+1} \neq 0$.

5 For $\delta = 1, 2$, u pushes $b_\tau(u_\delta)$ in $S_{i^*}^\delta$.
(Note that, if $C_{i^*+1}, \ldots, C_\tau$ are synchronized, then all agents are displaying the bit with index C_{i^*+1} of (C_{i^*}, b_{i^*}) as b_τ.)

6 While $i > 1$ and $C_i = 0$, u does the following:

7 | Pops the last $\ell_{i-1} - 1$ bits from S_{i-1}^δ and set s^δ equal to it.

8 | Sets C_{i-1} equal to the bitwise majority of $C_{i-1}(u)$, s^1 and s^2.

9 | Increments C_{i-1} and decrement i by one unit.

Algorithm 2: Iterative version of the protocol SYN-INTERMEDIATE, executed by each agent u, unfolding the recursion in proof of Lemma 5.7.

and we obtain a message size ℓ_i and a running time L_i, such that

$$L_{i+1} \leq \gamma_1(\log \ell_i \log n + \ell_i L_i), \tag{5.1}$$

for some constant γ_1 independent of i. Let a and b be two given numbers. Given two real numbers a and b, we use the notation $a \vee b$ to denote the maximum of a and b. Set L_1 to be $L_1 := L_{\text{SYN-SIMPLE}} \vee \log n = \mathcal{O}(\log T \log n) \vee \log n$, taking the maximum with $\log n$ for technical convenience. The second term dominates in Equation (5.1) because $\ell_i >> \log \ell_i$ and $L_i > \log n$. Hence, we obtain from Equation (5.1) that $L_{i+1} \leq 2\gamma_1 \ell_i L_i$. It follows by induction that

$$L_{i+1} \leq (2\gamma_1)^i L_1 \prod_{j=1}^{i} \ell_j.$$

The running time of EMUL(P) =SYN-CLOCK after the last application of the Message Reduction Theorem, i.e. τ, is thus

$$L_{\text{SYN-CLOCK}} := L_\tau \leq (2\gamma_1)^\tau L_1 \prod_{i=1}^{\tau-1} \ell_i. \tag{5.2}$$

To complete the estimate of the runtime, we use the following fact and Lemma 5.8.

Fact 5.4.2. *If $|x| < 1$, it holds*

$$e^{\frac{x}{1+x}} \leq 1 + x \leq e^x \leq 1 + \frac{x}{1-x}.$$

Lemma 5.8

Let $f, \tau : \mathbb{R}_+ \to \mathbb{R}$ be functions defined by $f(x) = \lceil \log x \rceil + 1$ and

$$\tau(x) = \inf \left\{ k \in \mathbb{N} \mid f^{\circledast k}(x) \leq 3 \right\}, \tag{5.3}$$

where we denote by $f^{\circledast k}$ the k-fold iteration of f. It holds that

$$\tau(T) \leq \log^{\circledast 4} T + \mathcal{O}(1). \tag{5.4}$$

Proof. We can notice that $f(T) \leq T - 1$, if T is bigger than some constant c. Moreover, when $f(x) \leq c$, the number of iterations before reaching 1 is $O(1)$. This implies that $\tau(T) \leq T + \mathcal{O}(1)$. But in fact, by definition, $\ell(T) = \tau\left(f^{\circledast 4}(T)\right) + 4$ (provided $f^{\circledast 4}(T) > 1$, which holds if T is big enough). Hence

$$\tau(T) \leq g\left(f^{\circledast 4}(T)\right) + 4 \leq f^{\circledast 4}(T) + \mathcal{O}(1) \leq \log^{\circledast 4} T + \mathcal{O}(1). \tag{5.5}$$

□

From the bounds $L_1 = \mathcal{O}(\log T \log n)$, $\prod_{i=1}^{\tau} \ell_i \leq \ell_1 \ell_2 \ell_3^\tau$, $\ell_1 = \mathcal{O}(\log T)$, $\ell_2 = \mathcal{O}(\log \log T)$ and Lemma 5.8, we obtain $(2\gamma_1)^\tau = (\log \log \log T)^{\mathcal{O}(1)} = \mathcal{O}(\log \log T)$ and

$$\ell_3^\tau \leq 2^{\mathcal{O}\left((\log \log \log \log T)^2\right)} \leq 2^{\mathcal{O}(\log \log \log T)} \leq (\log \log T)^{\mathcal{O}(1)}.$$

We thus conclude that

$$L_{\text{SYN-CLOCK}} \leq (2\gamma_1)^\tau \prod_{i=1}^{\tau} \ell_i L_1 \leq \mathcal{O}(\log \log T) \cdot \ell_1 \ell_2 \ell_3^\tau \cdot \mathcal{O}(\log T \log n) \tag{5.6}$$

$$\leq \mathcal{O}(\log \log T) \cdot \mathcal{O}(\log T) \cdot \mathcal{O}(\log \log T) \cdot \mathcal{O}(\log \log T)^{\mathcal{O}(1)} \cdot \mathcal{O}(\log T \log n) \tag{5.7}$$

$$\leq \log^2 T \log n \cdot (\log \log T)^{\mathcal{O}(1)}. \tag{5.8}$$

The total slowdown with respect to Syn-Simple corresponds to $\prod_{i=1}^{\tau} \ell_i = \tilde{\mathcal{O}}(\log T)$. Hence the clock produced by the emulation is incremented every $\tilde{\mathcal{O}}(\log T)$ rounds. In other words we obtain a clock modulo $T \cdot f(T)$ for some function f. But using Remark 5.4.1 we can still view this as a clock modulo T. □

Extension to General T and Running Time Improvement. We now aim to get rid of the assumption that T is a power of 2 in Lemma 5.7, and also reduce the running time of our protocol to $\tilde{\mathcal{O}}(\log n \log T)$, proving Theorem 5.2.

Syn-Clock protocol

Memory: Each agent u stores a clock $C'(u)$ which runs modulo $T' \gg \gamma \log n \log T$. Each agent u also stores a variable Q which is incremented only once every T' rounds and runs modulo T.

Message: Each agent u displays 4 bits. On the first 3 bits, protocol Syn-Intermediate is applied to synchronize C'. The 4-th bit $b(u)$ is the bit with index $(\lfloor \frac{C'(u)}{\gamma \log n} \rfloor \mod \lceil \log T \rceil)$ of $Q(u)$.

1 u samples two agents u_1 and u_2.

2 u updates $b(u)$ with the majority of $b(u)$, $b(u_1)$ and $b(u_2)$.

3 If $C' = 0$, increment Q by one unit modulo T.

Output: The clock modulo T is obtained as $C := (C' + Q \cdot T') \mod T$

Algorithm 3: The protocol 4-bit Syn-Clock, executed by each agent u.

Proof of Theorem 5.2. From Lemma 5.7, we know that Syn-Intermediate synchronizes clocks modulo T in time $\tilde{\mathcal{O}}\left(\log^2 T \log n\right)$ using only 3-bit messages, provided that T is a power of 2. While protocol Syn-Intermediate emulates protocol Syn-Simple, it displays the first bit of the message of Syn-Simple only once every $\tilde{\mathcal{O}}(\log T)$ rounds. Of course, it would be more efficient to display it $\mathcal{O}(\log n)$ times in a row, so that maj-consensus would make every agent agree on this bit, and then move to agreeing on the second bit, and so on. To achieve this, as in the proof of Syn-Simple, we can view a clock modulo T, say Q, as written on $\log T$ bits. If agents already possess a "small" counter modulo $T' := \mathcal{O}(\log T \log n)$ they can use it to display the first bit for $\mathcal{O}(\log n)$ rounds, then the second one for $\mathcal{O}(\log n)$ rounds, and so on until each one of the $\lceil \log T \rceil$ bits of T has been synchronized. This would synchronize all bits of the desired clock within $\mathcal{O}(\log T \log n)$ rounds, w.h.p., while being very economical in terms of message length, since only 1 bit is displayed at any time.

Therefore, we can use Lemma 5.7 to synchronize a counter modulo $\mathcal{O}(\log T \log n)$ in $\tilde{\mathcal{O}}((\log \log T)^2 \log n)$ rounds, using 3 bits per message. Then, we can use a fourth bit to run maj-consensus on each of the $\log T$ bits of Q for $\mathcal{O}(\log n)$ consecutive rounds, for a total running time of $\mathcal{O}(\log T \log n)$ rounds. At this point, an application of the Message Reduc-

tion Theorem would give us a protocol with running time $\mathcal{O}(\log T \log n)$ using 3-bit messages. However, perhaps surprisingly, a similar strategy enables us to synchronize a clock modulo any integer (not necessarily a power of 2).

Let us assume that $T \in \mathbb{N}$ is an arbitrary integer. Let $\gamma \log n$ be an upper bound on the convergence time of MAJ-CONSENSUS which guarantees a correct consensus with probability at least $1 - n^{-2}$, for some constant γ large enough [48]. Let T' be the smallest power of 2 bigger than $\log T \cdot (\gamma \log n + \gamma \log \log T)$. By Lemma 5.7, using 3 bits, the agents can build a synchronized clock C' running modulo T' in time $\tilde{\mathcal{O}}((\log \log T)^2 \log n)$. The other main ingredient in this construction is another clock $Q_{T'}$ which is incremented once every T' rounds and runs modulo T. The desired clock modulo T, which we denote C, is obtained by

$$C := \left(C' + Q_{T'} \cdot T'\right) \mod T. \tag{5.9}$$

It is easy to check, given the definitions of C' and $Q_{T'}$ that this choice indeed produces a clock modulo T.

It remains to show how the clock $Q_{T'}$ modulo T is synchronized. On a first glance, it may seem as if we did not simplify the problem since Q is a clock modulo T itself. However, the difference between $Q_{T'}$ and a regular clock modulo T is that $Q_{T'}$ is incremented only once every T' rounds. This is exploited as follows.

The counter $Q_{T'}$ is written on $\lceil \log T \rceil$ internal bits. We show how to synchronize $Q_{T'}$ using a 4-th bit in the messages, similarly to the aforementioned strategy to synchronize Q; we later show how to remove this assumption using the Message Reduction Theorem. Let us call a loop of C' modulo T' an *epoch*. The rounds of an epoch are divided in phases of equal length $\gamma \log n + \gamma \log \log T$ (the remaining $T' \mod (\gamma \log n + \gamma \log \log T)$ rounds are just ignored). The clock C' determines which bit from $Q_{T'}$ to display. The first bit of $Q_{T'}$ is displayed during the first phase, then the second one is displayed during the second phase, and so on. By Theorem 5.3, the length of each phase guarantees that consensus is achieved on each bit of $Q_{T'}$ via[5] MAJ-CONSENSUS w.h.p. More precisely, after the first bit has been displayed for $\gamma \log n + \gamma \log \log T$ rounds, all agents agree on it with probability[6] $1 - \frac{1}{n^2 \log T}$, provided γ is large enough. Thus, at the end of an epoch, agents agree on all $\lceil \log T \rceil$ bits of $Q_{T'}$ with probability greater than $(1 - \frac{1}{n^2 \log T})^{\log T} \gg 1 - \mathcal{O}(n^{-2})$.

We have thus shown that, by the time C' reaches its maximum value of T', i.e. after one epoch, all agents agree on $Q_{T'}$ w.h.p. and then increment it jointly. From Lemma 5.7,

[5] Observe that, once clock C' is synchronized, the bits of $Q_{T'}$ do not change for each agent during each subphase. Thus, we may replace MAJ-CONSENSUS by the MIN protocol where on each round of subphase i each agent u pulls another agent v u.a.r. and updates her i-th bit of Q to the minimum between her current i-th bit of Q and the one of v. However, for simplicity's sake, we reuse the already introduced MAJ-CONSENSUS protocol.

[6] From Theorem 5.3, we have that after $\gamma \log n$ rounds, with γ large enough, the probability that consensus has not been reached is smaller than $\frac{1}{n^2}$. Thus, after $N \cdot \gamma \log n$ rounds, the probability that consensus has not been reached is smaller than $\frac{1}{n^{2N}}$. If we choose $N \log n = \log n + \log \log T$, we thus get the claimed upper bound $\frac{1}{n^2 \log T}$.

Syn-Intermediate takes

$$\tilde{\mathcal{O}}\left(\log^2 T' \log n\right) = \mathcal{O}\left((\log\log n + \log\log T)^2 \log n\right) = \mathcal{O}\left(\left((\log\log n)^2 \log n + (\log\log T)^2 \log n\right)\right)$$

rounds to synchronize a clock C' modulo T' w.h.p. Together with the $\log T\,(\gamma \log n + \gamma \log\log T)$ rounds to agree on $Q_{T'}$ w.h.p., this implies that after $\log T \log n \cdot (\log\log T)^{\mathcal{O}(1)} \cdot (\log\log n)^{\mathcal{O}(1)} = \tilde{\mathcal{O}}(\log T \log n)$ rounds the clocks C are all synchronized w.h.p.

Finally, we show how to get rid of the extra 4-th bit to achieve agreement on $Q_{T'}$. Observe that, once C' is synchronized, this bit is used in a self-stabilizing way. Thus, since Syn-Intermediate has the bitwise-independence property, using Lemma 5.4, the protocol we described above possesses the bitwise-independence property too. By using the Message Reduction Theorem we can thus reduce the message size from 4 bits to $\lceil \log 4 \rceil + 1 = 3$ bits, while only incurring a constant multiplicative loss in the running time. The clock we obtain, counts modulo T but is incremented every 4 rounds only. It follows from Remark 5.4.1 that we may still view this as a clock modulo T. □

Remark 5.4.3 (Internal memory space)**.** *The internal memory space needed to implement our protocols* Syn-Simple*,* Syn-Intermediate*, and* Syn-Clock *is close to* $\log T$ *in all cases: protocol* Syn-Simple *uses one counter written on* $\log T$ *bits,* Syn-Intermediate *needs internal memory of size*

$$\log T + \mathcal{O}\left(\log\log T + \log\log\log T + \ldots\right) \leq \log T(1 + o(1)), \tag{5.10}$$

and the internal memory requirement of Syn-Clock *is of order* $\log T + \log\log n$.

5.5 Majority Bit Dissemination with a Clock

In this section we assume that agents are equipped with a synchronized clock C modulo $\gamma \log n$ for some big enough constant $\gamma > 0$. In the previous section we showed how to establish such a synchronized clock in $\tilde{\mathcal{O}}(\log n)$ time and using 3-bit messages. We have already seen in Section 5.1.2 how to solve the Bit Dissemination problem (when we are promised to have a single source agent) assuming such synchronized clocks, by paying an extra bit in the message size and an $\mathcal{O}(\log n)$ additive factor in the running time. This section is dedicated to showing that, in fact, the more general Majority Bit Dissemination problem can be solved with the same time complexity and using 3-bit messages, proving Theorem 5.1.

In Section 5.5.1, we describe and analyze protocol Syn-Phase-Spread, which solves Majority Bit Dissemination by paying only a $\mathcal{O}(\log n)$ additive overhead in the running time w.r.t. Clock Synchronization. For clarity's sake, we first assume that the protocol is using 4 bits (i.e. 1 additional bit over the 3 bits used for Clock Synchronization), and we later show how to decrease the number of bits back to 3 in Section 5.5.2, by applying the Message Reduction Theorem.

The main idea behind the 3(+1)-bit protocol, called Syn-Phase-Spread, is to make the sources' input bits disseminate on the system in a way that preserves the initial ratio $\frac{k_1}{k_0}$ between the number of sources supporting the majority and minority input bit. This is achieved by dividing the dissemination process in phases, similarly to the main protocol in [63] which was designed to solve the Bit Dissemination problem in a variant of the $\mathcal{PUSH}$ model in which messages are affected by noise. The phases induce a spreading process which allows to leverage on the concentration property of the Chernoff bounds, preserving the aforementioned ratio. While, on an intuitive level, the role of noisy messages in the model considered in [63] may be related to the presence of sources having conflicting opinion in our setting, we remark that our protocol and its analysis depart from those of [63] on several key points: while the protocol in [63] needs to know the noise parameter, Syn-Phase-Spread does not assume any knowledge about the number of different sources, and our analysis does not require to control the growth of the number of speaking agents from above.

In order to perform such spreading process with 1 bit only, the protocol in [63] leverages on the fact that in the $\mathcal{PUSH}$ model agents can choose *when to speak*, i.e. whether to send a message or not. To emulate this property in the $\mathcal{PULL}$ model, we use the parity of the clock C: on odd rounds agents willing to "send" a 0 display 0, while others display 1 and conversely on even rounds. Rounds are then grouped by two, so 2 rounds in the $\mathcal{PULL}$ model correspond to 1 round in the $\mathcal{PUSH}$ version.

5.5.1 Protocol Syn-Phase-Spread

In this section we describe protocol Syn-Phase-Spread. As mentioned above, for clarity's sake we assume that Syn-Phase-Spread uses 4-bit messages, and we show how to remove this assumption in Section 5.5.2. Three of such bits are devoted to the execution Syn-Clock, in order to synchronize a clock C modulo $2\lceil\gamma_{phase}\log n\rceil + \gamma_{phase}\lceil 2\log n\rceil$ for some constant γ_{phase} large enough. Throughout this section we assume, thanks to Theorem 5.2, that C has already been synchronized, which happens after $\tilde{\mathcal{O}}(\log n)$ rounds from the start of the protocol. In Section 5.5.1, we present a protocol Phase-Spread solving Majority Bit Dissemination assuming agents already share a common clock.

Protocol Phase-Spread

Let γ_{phase} be a constant to be set later. Protocol Phase-Spread is executed periodically over periods of length $2\lceil\gamma_{phase}\log n\rceil + \gamma_{phase}\lceil 2\log n\rceil$, given by a clock C. One run of length $2\lceil\gamma_{phase}\log n\rceil + \gamma_{phase}\lceil 2\log n\rceil$ is divided in $2 + \lceil 2\log n\rceil$ phases, the first and the last ones lasting $\lceil\gamma_{phase}\log n\rceil$ rounds, all the other $\lceil 2\log n\rceil$ phases lasting γ_{phase} rounds. The first phase is called *boosting*, the last one is called *polling*, and all the intermediate ones are called *spreading*. For technical convenience, in Phase-Spread agents disregard the messages they get as their second pull. In other words, Phase-Spread works in the $\mathcal{PULL}(1)$ model.

During the boosting and the spreading phases, as we already explained in the introduction

of this section, we make use of the parity of time to emulate the ability to actively send a message or not to communicate anything as in the $\mathcal{PUSH}$ model(in the first case we say that the agent is *speaking*, in the second case we say that the agent is *silent*). This induces a factor 2 slowdown which we henceforth omit for simplicity.

At the beginning of the boosting, each non-source agent u is silent. During the boosting and during each spreading phase, each silent agent pulls until she sees a speaking agent. When a silent agent u sees a speaking agent v, u memorizes $b_1(v)$ but remains silent until the end of the phase; at the end of the current phase, u starts speaking and sets $b_1(u) = b_1(v)$. The bit b_1 is then never modified until the clock C reaches 0 again. Then, during the polling phase, each agent u counts how many agents with $b_1 = 1$ and how many with $b_1 = 0$ she sees. At the end of the phase, each agent u sets their output bit to the most frequent value of b_1 observed during the polling phase. We want to show that, for all agents, the latter is w.h.p. b_{maj} (the most frequent initial opinion among sources).

Phase-Spread protocol

1 If u is not speaking and $b_1(u)$ has not yet been set, and the current phase is either the boosting or the spreading one, u does the following:

2 | u observes a random agent v.

3 | If v is speaking, u sets $b_1(u)$ equal to $b_1(v)$,
and u will be speaking from the next phase.

4 | u sets c_0 and c_1 equal to 0.

5 If the current phase is polling:

6 | u observes a random agent v.

7 | If $b_1(v) = 1$, u increments c_1, otherwise increment c_0.

8 u outputs 1 if and only if $c_1 > c_0$.

Algorithm 4: The protocol PHASE-SPREAD, executed by each agent u.

A Technical Tool

Before we can analyze the protocol, let us recall a large deviation lemma

Theorem 5.9: [100]

Let $X_1, ..., X_n$ be n independent random variables. If $X_i \leq M$ for each i, then

$$P\left(\sum_i X_i \geq \mathbb{E}\left[\sum_i X_i\right] + \lambda\right) \leq e^{-\frac{\lambda^2}{2\left(\sqrt{\sum_i \mathbb{E}\left[X_i^2\right]} + \frac{M\lambda}{3}\right)}}. \tag{5.11}$$

Corollary 5.10

Let $\mu = \mathbb{E}\left[\sum_i X_i\right]$. If the X_is are binary then, for $\lambda = \sqrt{\mu \log n}$ and sufficiently large n, (5.11) gives

$$P\left(\sum_i X_i \geq \mu + \sqrt{\mu \log n}\right) \leq e^{-\sqrt{\mu \log n}}, \tag{5.12}$$

$$P\left(\sum_i X_i \leq \mu - \sqrt{\mu \log n}\right) \leq e^{-\sqrt{\mu \log n}}. \tag{5.13}$$

Proof. The fact that the X_is are binary implies that $\sum_i \mathbb{E}\left[X_i^2\right] \leq \sum_i \mathbb{E}\left[X_i\right]$. By setting $\lambda = \sqrt{\mathbb{E}\left[\sum_i X_i\right] \log n}$, one can show that the l.h.s. of (5.11) is upper bounded by $e^{-\sqrt{\mu \log n}}$. □

Analysis

We prove that at the end of the last spreading phase w.h.p. all agents are speaking and each agent has $b_1 = 1$ with probability $\frac{1}{2} + \varepsilon_{end}$ for some positive constant $\varepsilon_{end} = \varepsilon_{end}(\gamma_{phase}, \varepsilon)$ (where the dependency in γ_{phase} is monotonically increasing), $b_1 = 0$ otherwise. From the Chernoff bound (Corollary 5.10) and the union bound, this implies that when $\gamma_{phase} > \frac{8}{\varepsilon_{end}}$ at the end of the polling phase w.h.p. each agent learns b_{maj}. Without loss of generality, let $b_{maj} = 1$, i.e. $k_1 > k_0$. For convenience, we estimate ratios of the form $\frac{k_1}{k_0}$, which requires that $k_0 > 0$. The analysis can easily be adapted to handle the case where $k_0 = 0$.

For $\varepsilon \in \{0, 1\}$, let us denote $k_\varepsilon^{(i)}$ the number of nodes with $b_1(\cdot) = \varepsilon$ at the end of phase i. The analysis is divided in the following lemmas.

Lemma 5.11

At the end of the boosting phase it holds w.h.p.

$$k_1^{(1)} + k_0^{(1)} \geq \begin{cases} (k_1+k_0)\,\frac{\gamma_{phase}}{3}\log n \text{ if } k_1+k_0 < \frac{n}{2\gamma_{phase}\log n} \\ n\left(1-\frac{1}{\sqrt{e}}\right) + \frac{1}{\sqrt{e}}\,(k_1+k_0) - \sqrt{n\log n} \\ \qquad \text{if } \frac{n}{2\gamma_{phase}\log n} \leq k_1+k_0 \leq n - 2\sqrt{n\log n}, \\ n \text{ otherwise.} \end{cases} \tag{5.14}$$

Moreover,

$$\frac{k_1^{(1)}}{k_0^{(1)}} \geq \frac{k_1}{k_0} \cdot \left(1 - \sqrt{\frac{9}{\gamma_{phase} k_0}}\right). \tag{5.15}$$

Proof. By using Fact 5.4.2, we have

$$\begin{aligned} \mathbb{E}\left[k_1^{(1)} + k_0^{(1)}\right] &= k_1 + k_0 + (n - k_1 - k_0)\left(1 - \left(1 - \frac{k_1+k_0}{n}\right)^{\gamma_{phase}\log n}\right) \\ &\geq k_1 + k_0 + (n - k_1 - k_0)\left(1 - e^{-\frac{k_1+k_0}{n}\gamma_{phase}\log n}\right). \end{aligned} \tag{5.16}$$

We distinguish three cases.

Case $k_1 + k_0 < \frac{n}{2\gamma_{phase}\log n}$. By using Fact 5.4.2 again, from (5.16) we get

$$\begin{aligned} \mathbb{E}\left[k_1^{(1)} + k_0^{(1)}\right] &\geq k_1 + k_0 + (n - k_1 - k_0)\left(1 - e^{-\frac{k_1+k_0}{n}\gamma_{phase}\log n}\right) \\ &\geq k_1 + k_0 + (n - k_1 - k_0)\,\frac{\frac{k_1+k_0}{n}\gamma_{phase}\log n}{1 + \frac{k_1+k_0}{n}\gamma_{phase}\log n} \\ &\geq k_1 + k_0 + (n - k_1 - k_0)\,\frac{k_1+k_0}{n}\,\frac{\gamma_{phase}}{2}\log n \\ &\geq k_1 + k_0 + \left(1 - \frac{k_1+k_0}{2n}\right)(k_1+k_0)\,\frac{\gamma_{phase}}{2}\log n \\ &\geq (k_1+k_0)\left(1 + \left(1 - \frac{1}{4\gamma_{phase}\log n}\right)\frac{\gamma_{phase}}{2}\log n\right) \qquad (5.17) \\ &\geq (k_1+k_0)\,\frac{\gamma_{phase}}{2}\log n. \qquad (5.18) \end{aligned}$$

From the Chernoff bound (Lemma 5.9), we thus get that w.h.p.

$$k_1^{(1)} + k_0^{(1)} \geq (k_1+k_0)\,\frac{\gamma_{phase}}{3}\log n.$$

Case $\frac{n}{2\gamma_{phase}\log n} \leq k_1 + k_0 \leq n - 2\sqrt{n\log n}$. From (5.16), we have

$$\mathbb{E}\left[k_1^{(1)} + k_0^{(1)}\right] \geq k_1 + k_0 + (n - k_1 - k_0)\left(1 - e^{-\frac{k_1+k_0}{n}\gamma_{phase}}\right) \tag{5.19}$$

$$\geq k_1 + k_0 + (n - k_1 - k_0)\left(1 - \frac{1}{\sqrt{e}}\right) \tag{5.20}$$

$$\geq n\left(1 - \frac{1}{\sqrt{e}}\right) + \frac{k_1 + k_0}{\sqrt{e}}. \tag{5.21}$$

From the Chernoff bound (Lemma 5.9), we thus get that w.h.p.

$$k_1^{(1)} + k_0^{(1)} \geq n\left(1 - \frac{1}{\sqrt{e}}\right) + \frac{k_1 + k_0}{\sqrt{e}} - \sqrt{n\log n}.$$

Case $k_1^{(1)} + k_0^{(1)} > n - 2\sqrt{n\log n}$. The probability that a silent agent does not observe a speaking one is

$$\left(\frac{n - k_1 - k_0}{n}\right)^{\gamma_{phase}\log n} \leq \left(\frac{4\log n}{n}\right)^{\frac{1}{2}\gamma_{phase}\log n},$$

hence by a simple union bound it follows that w.h.p. all agents are speaking.

Now, we prove (5.15). As before, we have two cases. The first case, $\frac{k_1}{k_0} \geq \frac{n}{2\gamma_{phase}\log n}$, is a simple consequence of the Chernoff bound (Lemma 5.9).

In the second case, $\frac{k_1}{k_0} < \frac{n}{2\gamma_{phase}\log n}$, let us consider the set of agents S_{boost} that start speaking at the end of the boosting, i.e. that observe a speaking agent during the phase. Observe that $|S_{boost}| = k_1^{(1)} - k_1 + k_0^{(1)} - k_0$. The probability that an agent in S_{boost} observes a source supporting 1 (resp. 0) is $\frac{k_1}{k_1+k_0}$ (resp. $\frac{k_0}{k_1+k_0}$). Thus

$$\begin{aligned}\mathbb{E}\left[k_1^{(1)}\right] &= k_1 + \frac{k_1}{k_1 + k_0}\mathbb{E}\left[|S_{boost}|\right] \quad \text{and} \\ \mathbb{E}\left[k_0^{(1)}\right] &= k_0 + \frac{k_0}{k_1 + k_0}\mathbb{E}\left[|S_{boost}|\right].\end{aligned} \tag{5.22}$$

In particular

$$\frac{\mathbb{E}\left[k_1^{(1)}\right]}{\mathbb{E}\left[k_0^{(1)}\right]} = \frac{k_1 + \frac{k_1}{k_1+k_0}\mathbb{E}\left[|S_{boost}|\right]}{k_0 + \frac{k_0}{k_1+k_0}\mathbb{E}\left[|S_{boost}|\right]} = \frac{k_1}{k_0}, \tag{5.23}$$

and from (5.18) and (5.22) we have

$$\mathbb{E}\left[k_0^{(1)}\right] \geq \frac{k_0}{k_1+k_0}\mathbb{E}\left[|S_{boost}|\right] \tag{5.24}$$

$$= \frac{k_0}{k_1+k_0}\left(\mathbb{E}\left[k_1^{(1)}+k_0^{(1)}\right] - (k_1+k_0)\right) \tag{5.25}$$

$$\geq (1-o(1))\frac{k_0}{k_1+k_0}\frac{\gamma_{phase}}{2}(k_1+k_0)\log n \tag{5.26}$$

$$= (1-o(1))k_0\frac{\gamma_{phase}}{2}\log n, \tag{5.27}$$

where the lower bound follows from the assumption $\frac{k_1}{k_0} < \frac{n}{2\gamma_{phase}\log n}$ and (5.18). From (5.27) and the multiplicative form of the Chernoff bound (Corollary 5.10), we have that w.h.p.

$$k_1^{(1)} \geq \mathbb{E}\left[k_1^{(1)}\right] - \sqrt{\mathbb{E}\left[k_1^{(1)}\right]\log n} \quad \text{and} \tag{5.28}$$

$$k_0^{(1)} \leq \mathbb{E}\left[k_0^{(1)}\right] + \sqrt{\mathbb{E}\left[k_0^{(1)}\right]\log n}. \tag{5.29}$$

Thus, since (5.22) implies $\mathbb{E}\left[k_1^{(1)}\right] \geq \mathbb{E}\left[k_0^{(1)}\right]$, we have

$$\frac{k_1^{(1)}}{k_0^{(1)}} \geq \frac{\mathbb{E}\left[k_1^{(1)}\right] - \sqrt{\mathbb{E}\left[k_1^{(1)}\right]\log n}}{\mathbb{E}\left[k_0^{(1)}\right] + \sqrt{\mathbb{E}\left[k_0^{(1)}\right]\log n}} \tag{5.30}$$

$$= \frac{\mathbb{E}\left[k_1^{(1)}\right]}{\mathbb{E}\left[k_0^{(1)}\right]} \cdot \frac{1-\sqrt{\frac{\log n}{\mathbb{E}\left[k_1^{(1)}\right]}}}{1+\sqrt{\frac{\log n}{\mathbb{E}\left[k_0^{(1)}\right]}}} \tag{5.31}$$

$$\geq \frac{\mathbb{E}\left[k_1^{(1)}\right]}{\mathbb{E}\left[k_0^{(1)}\right]} \cdot \left(1-\sqrt{\frac{\log n}{\mathbb{E}\left[k_1^{(1)}\right]}} - \sqrt{\frac{\log n}{\mathbb{E}\left[k_0^{(1)}\right]}}\right) \tag{5.32}$$

$$\geq \frac{\mathbb{E}\left[k_1^{(1)}\right]}{\mathbb{E}\left[k_0^{(1)}\right]} \cdot \left(1-2\sqrt{\frac{\log n}{\mathbb{E}\left[k_0^{(1)}\right]}}\right) \tag{5.33}$$

$$= \frac{k_1}{k_0} \cdot \left(1-\sqrt{\frac{9}{k_0\gamma_{phase}}}\right), \tag{5.34}$$

concluding the proof. □

Lemma 5.12

At the end of the $i+1$th spreading phase, the following holds w.h.p.

$$k_1^{(i+1)} + k_0^{(i+1)} \geq \begin{cases} \left(k_1^{(i)} + k_0^{(i)}\right) \frac{\gamma_{phase}}{3}, \text{ if } k_1^{(i)} + k_0^{(i)} < \frac{n}{2\gamma_{phase}} \\ n\left(1 - \frac{1}{\sqrt{e}}\right) + \frac{1}{\sqrt{e}}\left(k_1^{(i)} + k_0^{(i)}\right) - \sqrt{n \log n} \\ \qquad \text{if } \frac{n}{2\gamma_{phase}} \leq k_1^{(i)} + k_0^{(i)} \leq n - 2\sqrt{n \log n} \\ n, \text{ otherwise.} \end{cases}$$

$$\frac{k_1^{(i+1)}}{k_0^{(i+1)}} \geq \frac{k_1^{(i)}}{k_0^{(i)}} \left(1 - 4\sqrt{\frac{\log n}{\gamma_{phase} k_0^{(i)}}}\right). \tag{5.35}$$

Proof. The proof is almost the same as that of Lemma 5.11. Thus, we condense some analogous calculations.

By using Fact 5.4.2, we have

$$\mathbb{E}\left[k_1^{(i+1)} + k_0^{(i+1)}\right] \geq k_1^{(i)} + k_0^{(i)} + \left(n - k_1^{(i)} - k_0^{(i)}\right)\left(1 - e^{-\frac{k_1^{(i)} + k_0^{(i)}}{n}\gamma_{phase}}\right). \tag{5.36}$$

We distinguish three cases.

Case $k_1^{(i)} + k_0^{(i)} < \frac{n}{2\gamma_{phase}}$. By using Fact 5.4.2 again, from (5.36) we get

$$\mathbb{E}\left[k_1^{(i+1)} + k_0^{(i+1)}\right] \geq k_1^{(i)} + k_0^{(i)} + \left(n - k_1^{(i)} - k_0^{(i)}\right) \cdot \frac{k_1^{(i)} + k_0^{(i)}}{2n}\gamma_{phase} \geq \left(k_1^{(i)} + k_0^{(i)}\right)\frac{\gamma_{phase}}{2}. \tag{5.37}$$

After the boosting phase, i.e. for $i \geq 1$, it follows from Lemma 5.11 that $k_1^{(i)} + k_0^{(i)} = \Omega\left(\gamma_{phase} \log n\right)$. From the Chernoff bound (Lemma 5.9), if γ_{phase} is chosen big enough, we thus get that w.h.p.

$$k_1^{(i+1)} + k_0^{(i+1)} \geq \left(k_1^{(i)} + k_0^{(i)}\right)\frac{\gamma_{phase}}{3}.$$

Case $\frac{n}{2\gamma_{phase}} \leq k_1^{(i)} + k_0^{(i)} \leq n - 2\sqrt{n \log n}$. From (5.36), we have

$$\mathbb{E}\left[\left(k_1^{(i+1)} + k_0^{(i+1)}\right)\right] \geq k_1^{(i)} + k_0^{(i)} + \left(n - k_1^{(i)} - k_0^{(i)}\right)\left(1 - \frac{1}{\sqrt{e}}\right) \geq n\left(1 - \frac{1}{\sqrt{e}}\right) + \frac{1}{\sqrt{e}}\left(k_1^{(i)} + k_0^{(i)}\right). \tag{5.38}$$

From the Chernoff bound (Lemma 5.9), we thus get that w.h.p.

$$k_1^{(i+1)} + k_0^{(i+1)} \geq n\left(1 - \frac{1}{\sqrt{e}}\right) + \frac{1}{\sqrt{e}}\left(k_1^{(i)} + k_0^{(i)}\right) - \sqrt{n \log n}.$$

Case $k_1^{(i)} + k_0^{(i)} > n - 2\sqrt{n \log n}$. The probability that a silent agent does not observe a speaking one is

$$\left(\frac{n - k_1^{(i)} - k_0^{(i)}}{n}\right)^{\gamma_{phase}} \leq \left(\frac{4 \log n}{n}\right)^{\frac{1}{2}\gamma_{phase}},$$

hence by a simple union bound it follows that w.h.p. all agents are speaking.

Now, we prove (5.35). As in the proof of (5.15), we have two cases. The first case, $\frac{k_1}{k_0} \geq \frac{n}{2\gamma_{phase}}$, is a simple consequence of the Chernoff bound (Lemma 5.9). Otherwise, let us assume $\frac{k_1}{k_0} < \frac{n}{2\gamma_{phase}}$. With an analogous argument to that for (5.22) and (5.23) we can prove

$$\frac{\mathbb{E}\left[k_1^{(i+1)}\right]}{\mathbb{E}\left[k_0^{(i+1)}\right]} = \frac{k_1^{(i)}}{k_0^{(i)}}, \tag{5.39}$$

and

$$\mathbb{E}\left[k_1^{(i+1)}\right] = k_1^{(i)} + \frac{k_1^{(i)}}{k_1^{(i)} + k_1^{(i)}} \mathbb{E}\left[k_1^{(i+1)} - k_1^{(i)} + k_0^{(i+1)} - k_0^{(i)}\right], \tag{5.40}$$

$$\mathbb{E}\left[k_0^{(i+1)}\right] = k_0^{(i)} + \frac{k_0^{(i)}}{k_1^{(i)} + k_0^{(i)}} \mathbb{E}\left[k_1^{(i+1)} - k_1^{(i)} + k_0^{(i+1)} - k_0^{(i)}\right]. \tag{5.41}$$

As in (5.29), from the multiplicative form of the Chernoff bound (Corollary 5.10) we have that w.h.p.

$$k_1^{(i+1)} \geq \mathbb{E}\left[k_1^{(i+1)}\right] - \sqrt{\mathbb{E}\left[k_1^{(i+1)}\right] \log n} \quad \text{and} \tag{5.42}$$

$$k_0^{(i+1)} \leq \mathbb{E}\left[k_0^{(i+1)}\right] + \sqrt{\mathbb{E}\left[k_0^{(i+1)}\right] \log n}. \tag{5.43}$$

Thus, as in (5.34), from (5.43) and (5.39), we get

$$\frac{k_1^{(i+1)}}{k_0^{(i+1)}} \geq \frac{\mathbb{E}\left[k_1^{(i+1)}\right]}{\mathbb{E}\left[k_0^{(i+1)}\right]} \cdot \left(1 - 2\sqrt{\frac{\log n}{\mathbb{E}\left[k_0^{(i+1)}\right]}}\right) \geq \frac{k_1^{(i)}}{k_0^{(i)}} \cdot \left(1 - 4\sqrt{\frac{\log n}{\gamma_{phase} k_0^{(i)}}}\right), \tag{5.44}$$

where, as in (5.27), in the last inequality we used that from (5.37) and (5.41) it holds $\mathbb{E}\left[k_0^{(i+1)}\right] \geq \frac{\gamma_{phase}}{4} k_0^{(i)}$. □

From the previous two lemmas, we can derive the following corollary, which concludes the proof.

Corollary 5.13

If $k_1 \geq k_0(1+\varepsilon)$ for some constant $\varepsilon > 0$, then at the end of the last spreading phase it holds w.h.p.

$$k_1^{(1+2\log n)} = n - k_0^{(1+2\log n)} \geq k_0^{(1+2\log n)}\,(1+\varepsilon_{end})\,, \tag{5.45}$$

where $\varepsilon_{end} = \frac{\varepsilon}{2} - \frac{4}{\sqrt{\gamma_{phase}}}$.

Proof. The equality in (5.45) follows from the first part of Lemma 5.11. When $k_1^{(i)} + k_0^{(i)} < \frac{n}{2\gamma_{phase}}$, $k_1^{(i)} + k_0^{(i)}$ increases by multiplicative a factor γ_{phase} at the end of each spreading phase. When $\frac{n}{2\gamma_{phase}} \leq k_1^{(i)} + k_0^{(i)} \leq n - 2\sqrt{n\log n}$,

$$n - k_1^{(i+1)} - k_0^{(i+1)} \leq \frac{n - k_1^{(i)} - k_0^{(i)}}{\sqrt{e}} - \sqrt{n\log n} \leq \frac{n - k_1^{(i)} - k_0^{(i)}}{\sqrt{e}}. \tag{5.46}$$

Hence the number of silent agents decreases by a factor $\sqrt{e}$ after each spreading phase. Lastly, when $k_1^{(i)} + k_0^{(i)} > n - 2\sqrt{n\log n}$, after one more spreading phase, a simple application of the union bound shows that $k_1^{(i+1)} + k_0^{(i+1)}$ is equal to n w.h.p. As a consequence, if γ_{phase} is big enough, after less than $1 + 2\log n$ spreading phases w.h.p it holds that $k_1^{(1+2\log n)} = n - k_0^{(1+2\log n)}$.

The inequality in (5.45) can be derived from (5.35), as follows. From (5.15) and (5.35) we have

$$\frac{k_1^{(1+2\log n)}}{k_0^{(1+2\log n)}} \geq \frac{k_1}{k_0}\left(1 - \sqrt{\frac{9}{\gamma_{phase}k_0}}\right)\prod_{i=2}^{1+2\log n}\left(1 - \sqrt{\frac{16\log n}{\gamma_{phase}k_0^{(i)}}}\right). \tag{5.47}$$

We can estimate the product as

$$\begin{aligned}
\prod_{i=2}^{1+2\log n}\left(1 - \sqrt{\frac{16\log n}{\gamma_{phase}k_0^{(i)}}}\right) &\geq \exp\left(-4\sum_{i=2}^{1+2\log n}\frac{1}{\left(\sqrt{\gamma_{phase}}\right)^i}\right) \\
&\geq \exp\left\{-4\left(\frac{1}{\gamma_{phase} - \sqrt{\gamma_{phase}}} - n^{-\frac{2\log\gamma_{phase}}{2}}\right)\right\} \\
&\geq \exp\left\{-\frac{4}{\gamma_{phase}}\right\} \\
&\geq \left(1 - \frac{5}{\gamma_{phase}}\right).
\end{aligned} \tag{5.48}$$

In the first inequality we used that $1 - x \geq e^{-x}$ if $|x| < 1$. To go from the first to the second line, we use $\sum_2^a x = -1 - x + \frac{1-x^{a+1}}{1-x}$, with $a = 1 + 2\log n$ and $x = \frac{1}{\left(\sqrt{\gamma_{phase}}\right)^i}$. To go from the

third to the fourth line, we use that for n big enough, $\frac{1}{\gamma_{phase}-\sqrt{\gamma_{phase}}} - n^{-\frac{2\log\gamma_{phase}}{2}} \geq \frac{1}{\gamma_{phase}}$. The last inequality is obtained from $e^{-x} \geq 1 - \frac{5}{4}x$, if $x > 0$ is small enough.

Finally, from (5.47) and (5.48) we get

$$\frac{k_1^{(1+2\log n)}}{k_0^{(1+2\log n)}} \geq \frac{k_1}{k_0}\left(1-\sqrt{\frac{9}{\gamma_{phase}k_0}}\right)\left(1-\frac{5}{\gamma_{phase}}\right) \geq \frac{k_1}{k_0}\left(1-\frac{4}{\sqrt{\gamma_{phase}}}\right), \tag{5.49}$$

which, together with the hypothesis $\frac{k_1}{k_0} \geq 1+\varepsilon$, concludes the proof. □

5.5.2 Proof of Theorem 5.1

Proof. From Corollary 5.13, it follows that at the end of the last spreading phase, all agents have been informed. After the last spreading phase, during the polling phase, each agent samples $\gamma_{phase}\log n$ opinions from the population and then adopts the majority of these as her output bit. Thus, (5.45) ensures that each sample holds the correct opinion with probability $\geq \frac{1}{2}+\varepsilon_{end}$. Hence, by the Chernoff bound and a union bound, if γ_{phase} is big enough then the majority of the $\gamma_{phase}\log n$ samples corresponds to the correct value for all the n agents w.h.p.

The protocol obtained so far solves Majority Bit Dissemination, but it does it using 4 bits per message rather than 3. Indeed, synchronizing a clock using Syn-Clock takes 3 bits, and we use an extra bit to execute Phase-Spread described in Section 5.5.1. However, the protocol Syn-Phase-Spread has the bitwise-independence property. This follows from Lemma 5.4 with Syn-Generic =Syn-Clock, P =Phase-Spread, Syn-P =Syn-Phase-Spread, together with the observation that Phase-Spread is self-stabilizing. We can thus reduce the message length of Syn-Phase-Spread to 3 bits using again the Message Reduction Theorem, with a time overhead of a factor 4 only. □

5.6 Conclusion and Open Problems

This chapter deals with the construction of protocols in highly congested stochastic interaction patterns. Corresponding challenges are particularly evident when it is difficult to guarantee synchronization, which seems to be essential for emulating a typical protocol that relies on many bits per message with a protocol that uses fewer bits. We showed that in the $\mathcal{PULL}$ model, if a self-stabilizing protocol satisfies the bitwise-independence property then it can be emulated with only 3 bits per message. Using this rather general transformer, we solve the self-stabilizing Clock-Synchronization and Majority Bit Dissemination problems in almost-logarithmic time and using only 3 bits per message. It remains an open problem whether the message size of either one of these problems can be further reduced while keeping the running time polylogarithmic.

In particular, even for the self-stabilizing Bit Dissemination problem (with a single source) it remains open whether there exists a polylogarithmic protocol that uses a single bit per

interaction. In fact, we investigated several candidate protocols which seem promising in experimental simulation, but appear to be out of reach of current techniques for analysing randomly-interacting agent systems in a self-stabilizing context. Let us informally present one of them. Let ℓ,$k \in \mathbb{N}$ be two parameters. Agents can be in 3 states: *boosting*, *frozen* or *sensitive*. Boosting agents behave as in the MAJ-CONSENSUS protocol: they apply the majority rule to the 2 values they see in a given round and make it into their opinion for the next round. They also keep a counter T. If they have seen only agents of a given color b for ℓ rounds, they become sensitive to the opposite value. b-sensitive agents turn into frozen-b agents if they see value b. b-frozen agents keep the value b for k rounds before becoming boosters again. Intuitively what we expect is that, from every configuration, at some point almost all agents would be in the boosting state. Then, the boosting behavior would lead the agents to converge to a value b (which depends on the initial conditions). Most agents would then become sensitive to $1 - b$. If the source has opinion $1 - b$ then there should be a "switch" from b to $1 - b$. The "frozen" period is meant to allow for some delay in the times at which agents become sensitive, and then flip their opinion.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this thesis, we have presented two groups of works exploring new ways to combine algorithmic perspectives with biological experiments. The works presented in this manuscript are geared towards a computer science audience, but their inspiration comes from experiments on ants, which they also contributed to shape. The insights generated by our theoretical results lead to new interpretations of the biological systems they describe.

Our first theoretical contribution can be summarized as introducing and extensively studying the Noisy Advice model on trees. This model of graph search with permanent faults, captures some aspects of the collaborative transport process displayed by *Crazy Ants*. Studying it on trees is naturally appealing from a Theoretical Computer Science perspective, given the broad importance of trees in the discipline.

We provide tight results under different convergence requirements for move and query complexity. Specifically, we show that when the noise is low, smaller than $1/\sqrt{\Delta}$, it is possible to obtain strategies that find the target with an expected linear number of steps in the distance to the treasure and almost logarithmic number of queries in the number of nodes. Conversely, when the noise is of order greater than $\frac{1}{\sqrt{\Delta}}$, the expected number of steps (resp. queries) becomes exponential (resp. polynomial) in the distance to the target (resp. size of the tree). Working with high probability guarantees leads to different results. If we write the noise as $\Delta^{-\varepsilon}$ for some $\varepsilon > 0$, then it is possible to design strategies that walk no more than $d^{O(\varepsilon^{-1})}$ steps before finding the target, where d denotes the distance to the target. A similar result holds for queries. It is worth noting that in the high probability world, there is no threshold phenomenon as in the expectation case. Thus our works provide a natural model with a difference between these two kind of convergence requirements.

Our second theoretical contribution is extending our understanding of the conditions under which broadcast can be performed in $\mathcal{PULL}$-$\mathcal{PUSH}$ type of models. Such models are well suited to describe biological systems composed of computationally limited agents, interacting in a stochastic decentralized way. We first show in Section 4 a polynomial lower bound for

broadcast when interactions are noisy, and only pull interactions are allowed. In fact, it was shown previously by Feinerman & al. [63] that the same problem can be solved in logarithmic time if push interactions are allowed. Thus, our result provides an exponential separation between the $\mathcal{PUSH}$ and $\mathcal{PULL}$ models. These two models may look very close: in the absence of noise, it is possible to emulate uniform push interactions in a pull model. To do so, each agent adds a bit in their message indicating whether they are looking to communicate. However, noise may very well corrupt this extra bit in the $\mathcal{PULL}$ model. In the $\mathcal{PUSH}$ model, the very act of engaging in an interaction is not "noisy" and this is the crucial difference that allows to broadcast efficiently.

On more practical grounds, the high expressiveness of the lower bound does not prevent us from showing a close connection to particular experiments on a desert ant species called *Cataglyphis niger*. The polynomial lower bound is in accordance with the experimental data. In a nutshell, the broadcast time increases with group size. Another way to gain biological insights from our negative result is by considering its contrapositive. If a system is able to broadcast efficiently, it has to possess some form of structural stability (as opposed to randomized interactions), or be able to eliminate noise.

On the positive side we show an upper bound in Chapter 5, under noiseless pull interactions. This positive result is further enhanced by the self-stabilization property our protocol has and its very parsimonious message complexity. Indeed, only 3-bits per message are needed to secure efficient self-stabilizing broadcast, in that model. It is also possible to interpret the lower

6.2 Future Directions

At a general level, we hope the methodology we used in this thesis can be used and refined with other biological entities. It could be nice for instance to obtain a lower bound matching the performance of the system under consideration. Several technical questions remain to be studied in the models we considered, some were already mentioned at the end of each chapter. In the context of noisy advice, we posed the study of memoryless Probabilistic Following algorithms on general graphs as an intriguing open problem. It would also be interesting to obtain query or walk strategies for some specific graph families. Random permanent faults may also be relevant for other problems than search. They have already been further studied in the context of sorting [33].

Our broadcast lower bound could be extended in several ways. For instance, we could consider meeting patterns that are not fully random and different noise assumptions. As for our broadcast upper bound, it would be nice to obtain more *simple* self-stabilizing algorithms, in the sense for instance that they do not rely on a clock, or perhaps that they use even less than 3 bits in the messages. Several candidate protocols seem promising in experimental simulation, but their analysis seems beyond reach of current techniques. One such protocol was described at the end of Chapter 5. The paper by Dudek and Kosowski [54], provides a very interesting partial answer to this set of questions by giving some constant memory protocols, in a closely related variant of our model.

Bibliography

[1] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. *Distributed Computing*, 26(4):195–208, 2013.

[2] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-joseph. A biological solution to a fundamental distributed computing problem. *Science*, 2011.

[3] D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest. Time-space trade-offs in population protocols. *SODA*, pages 2560–2579, 2017.

[4] D. Alistarh, J. Aspnes, and R. Gelashvili. Space-optimal majority in population protocols. *SODA*, pages 2221–2239, 2018.

[5] N. Alon, M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler. Reliable communication over highly connected noisy networks. *PODC*, pages 165–173, 2016.

[6] F. Amor, P. Ortega, X. Cerdá, and R. Boulay. Cooperative prey-retrieving in the ant cataglyphis floricola: An unusual short-distance recruitment. *Insectes Sociaux*, 57(1), 2010.

[7] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[8] D. Angluin, J. Aspnes, and D. Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

[9] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *TAAS*, 3(4), 2008.

[10] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[11] J. A. Aslam and A. Dhagat. Searching in the presence of linearly bounded errors. *STOC*, pages 486–493, 1991.

[12] J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the EATCS*, 93:98–117, 2007.

[13] H. Attiya, A. Herzberg, and S. Rajsbaum. Optimal clock synchronization under different delay assumptions. *SIAM J. Comput.*, 25(2):369–389, 1996.

[14] F. Bartumeus and J. Catalan. Optimal search behavior and classic foraging theory. *Journal of Physics a-Mathematical and Theoretical*, 42, 10 2009.

[15] J. Beauquier, J. Burman, and S. Kutten. A self-stabilizing transformer for population protocols with covering. *Theor. Comput. Sci.*, 412(33):4247–4259, 2011.

[16] L. Becchetti, A. Clementi, E. Natale, F. Pasquale, and G. Posta. Self-stabilizing repeated balls-into-bins. *SPAA*, pages 332–339, 2015.

[17] L. Becchetti, A. E. F. Clementi, E. Natale, F. Pasquale, and R. Silvestri. Plurality consensus in the gossip model. *SODA*, pages 371–390, 2015.

[18] L. Becchetti, A. E. F. Clementi, E. Natale, F. Pasquale, and L. Trevisan. Stabilizing consensus with many opinions. *SODA*, pages 620–635, 2016.

[19] Y. Ben-Asher, E. Farchi, and I. Newman. Optimal search in trees. *SIAM J. Comput.*, 28(6):2090–2102, 1999.

[20] M. Ben-Or, D. Dolev, and E. N. Hoch. Fast self-stabilizing byzantine tolerant digital clock synchronization. *PODC*, pages 385–394, 2008.

[21] M. Ben-Or and A. Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). *FOCS*, pages 221–230, 2008.

[22] W. Bialek. Physical limits to sensation and perception. *Annual review of biophysics and biophysical chemistry*, 16(1):455–478, 1987.

[23] L. Boczkowski, A. Korman, and E. Natale. Minimizing message size in stochastic communication patterns: Fast self-stabilizing protocols with 3 bits. *SODA*, pages 2540–2559, 2017.

[24] L. Boczkowski, O. Feinerman, A. Korman, and E. Natale. Limits for rumor spreading in stochastic populations. *ITCS*, pages 49:1–49:21, 2018.

[25] L. Boczkowski, O. Feinerman, A. Korman, and E. Natale. Limits on reliable information flows through stochastic populations. *PLOS Computational Biology*, 2018.

[26] L. Boczkowski, B. Guinard, A. Korman, Z. Lotker, and M. Renault. Random walks with multiple step lengths. *LATIN*, pages 174–186, 2018.

[27] L. Boczkowski, I. Kerenidis, and F. Magniez. Streaming communication protocols. *ICALP*, pages 130:1–130:14, 2017.

[28] L. Boczkowski, A. Korman, and U. Feige. Typically fast search on trees with permanently noisy advice. *under submission*.

[29] L. Boczkowski, A. Korman, and E. Natale. Minimizing message size in stochastic communication patterns: Fast self-stabilizing protocols with 3 bits. *SODA*, pages 2540–2559, 2017.

[30] L. Boczkowski, A. Korman, and E. Natale. Minimizing message size in stochastic communication patterns: Fast self-stabilizing protocols with 3 bits. *Distributed Computing*, 2018.

[31] L. Boczkowski, A. Korman, and Y. Rodeh. Searching a tree with permanently noisy advice. *ESA*, 2018.

[32] R. S. Borgstrom and S. R. Kosaraju. Comparison-based search in the presence of errors. *STOC*, pages 130–136, 1993.

[33] M. Braverman and E. Mossel. Noisy sorting without resampling. *SODA*, pages 268–276, 2008.

[34] G. S. Brodal, R. Fagerberg, I. Finocchi, F. Grandoni, G. F. Italiano, A. G. Jørgensen, G. Moruz, and T. Mølhave. Optimal resilient dynamic dictionaries. *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 347–358, 2007.

[35] A. Cavagna, A. Cimarelli, I. Giardina, G. Parisi, R. Santagati, F. Stefanini, and M. Viale. Scale-free correlations in starling flocks. *PNAS*, 107(26):11865–11870, 2010.

[36] K. Censor-Hillel, B. Haeupler, J. A. Kelner, and P. Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. *STOC*, pages 961–970, 2012.

[37] E. Chastain, A. Livnat, C. Papadimitriou, and U. Vazirani. Algorithms, games, and evolution. *PNAS*, 111(29):10620–10623, 2014.

[38] B. Chazelle. Natural algorithms. *SODA*, pages 422–431, 2009.

[39] B. Chazelle. The convergence of bird flocking. *J. ACM*, 61(4):21:1–21:35, 2014.

[40] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumor spreading in social networks. *ICALP*, pages 375–386, 2009.

[41] F. Cicalese and U. Vaccaro. Optimal strategies against a liar. *Theor. Comput. Sci.*, 230(1-2):167–193, 2000.

[42] C. Cooper, R. Elsässer, T. Radzik, N. Rivera, and T. Shiraga. Fast consensus for voting on general expander graphs. *DISC*, pages 248–262, 2015.

[43] I. Couzin, J. Krause, N. Franks, and S. Levin. Effective leadership and decision making in animal groups on the move. *Nature 433*, pages 513–516, 2005.

[44] T. M. Cover and B. Gopinath. *Open problems in communication and computation.* Springer Science & Business Media, 2012.

[45] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. *PODC*, 1987.

[46] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[47] B. Doerr and M. Fouz. Asymptotically optimal randomized rumor spreading. *Electronic Notes in Discrete Mathematics*, 38:297–302, 2011.

[48] B. Doerr, L. A. Goldberg, L. Minder, T. Sauerwald, and C. Scheideler. Stabilizing consensus with the power of two choices. *SPAA*, pages 149–158, 2011.

[49] D. Dolev and E. N. Hoch. On self-stabilizing synchronous actions despite byzantine attacks. *DISC*, pages 193–207, 2007.

[50] S. Dolev. Possible and impossible self-stabilizing digital clock synchronization in general graphs. *Real-Time Systems*, 12(1):95–107, 1997.

[51] S. Dolev and J. L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51(5):780–799, 2004.

[52] A. Dornhaus and L. Chittka. Food alert in bumblebees (bombus terrestris): Possible mechanisms and evolutionary implications. *Behavioral Ecology and Sociobiology*, 50(6):570–576, 2001.

[53] A. Drewitz and A. F. Ramiréz. Selected topics in random walk in random environment. *Topics in Percolative and Disordered Systems, Springer Proceedings in Mathematics and Statistics*, 69:23–83, 2014.

[54] B. Dudek and A. Kosowski. Universal protocols for information dissemination using emergent signals. *STOC*, pages 87–99, 2018.

[55] R. Elsässer, T. Friedetzky, D. Kaaser, F. Mallmann-Trenn, and H. Trinker. Efficient k-party voting with two choices. *CoRR*, abs/1602.04667, 2016.

[56] R. Elsässer and T. Sauerwald. On the runtime and robustness of randomized broadcasting. *Theor. Comput. Sci.*, 410(36):3414–3427, 2009.

[57] E. Emamjomeh-Zadeh, D. Kempe, and V. Singhal. Deterministic and probabilistic binary search in graphs. *STOC*, pages 519–532, 2016.

[58] Y. Emek and R. Wattenhofer. Stone age distributed computing. *PODC*, pages 137–146, 2013.

[59] Y. Emek, T. Langner, D. Stolz, J. Uitto, and R. Wattenhofer. How many ants does it take to find the food? *SIROCCO*, pages 263–278, 2014.

[60] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Solving the ANTS problem with asynchronous finite state machines. *ICALP*, pages 471–482, 2014.

[61] J. M. Emlen. The role of time and energy in food preference. *The American Naturalist*, 100(916):611–617, 1966.

[62] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, October 1994.

[63] O. Feinerman, B. Haeupler, and A. Korman. Breathe before speaking: efficient information dissemination despite noisy, limited and anonymous communication. *PODC, 2014*.

[64] O. Feinerman and A. Korman. Theoretical distributed computing meets biology: A review. *ICDCIT*, pages 1–18, 2013.

[65] O. Feinerman and A. Korman. Clock synchronization and estimation in highly dynamic networks: An information theoretic approach. *SIROCCO*, pages 16–30, 2015.

[66] O. Feinerman and A. Korman. Individual versus collective cognition in social insects. *Submitted to Journal of Experimental Biology*, 2016.

[67] O. Feinerman, A. Rotem, and E. Moses. Reliable neuronal logic devices from patterned hippocampal cultures. *Nature physics*, 4(12):967–973, 2008.

[68] O. Feinerman and A. Korman. The ANTS problem. *Distributed Computing*, 30(3):149–168, 2017.

[69] I. Finocchi, F. Grandoni, and G. F. Italiano. Resilient search trees. *SODA*, pages 547–553, 2007.

[70] I. Finocchi and G. F. Italiano. Sorting and searching in the presence of memory faults (without redundancy). *STOC*, pages 101–110, 2004.

[71] E. Fonio, Y. Heyman, L. Boczkowski, A. Gelblum, A. Kosowski, A. Korman, and O. Feinerman. A locally-blazed ant trail achieves efficient collective navigation despite limited information, eLife 2016;5:e20185. 2016.

[72] P. Fraigniaud and E. Natale. Noisy rumor spreading and plurality consensus. *PODC*, pages 127–136, 2016.

[73] R. G. Gallager. Finding parity in a simple broadcast network. *IEEE Trans. Inf. Theor.*, 34(2):176–180, 2006.

[74] L. Gasieniec and G. Stachowiak. Fast space optimal leader election in population protocols. *SODA*, pages 2653–2667, 2018.

[75] A. Gelblum, I. Pinkoviezky, E. Fonio, A. Ghosh, and O. Feinerman. Ant groups optimally amplify the effect of transiently informed individuals. *Nature Communications*, 6:7729, 07 2015.

[76] L. A. Giraldeau, T. J. Valone, and J. Templeton. Potential disadvantages of using socially acquired information. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 357(1427):1559–1566, 2002.

[77] N. Goyal, G. Kindler, and M. E. Saks. Lower bounds for the noisy broadcast problem. *SIAM J. Comput.*, 37(6):1806–1841, 2008.

[78] R. H. MacArthur and E. R. Pianka. On optimal use of a patchy environment. *The American Naturalist*, 100:603–609, 01 1966.

[79] N. Hanusse, D. Ilcinkas, A. Kosowski, and N. Nisse. Locating a target with an agent guided by unreliable local advice: How to beat the random walk when you have a clock? *PODC*, pages 355–364, 2010.

[80] N. Hanusse, D. Kavvadias, E. Kranakis, and D. Krizanc. Memoryless search algorithms in a network with faulty advice. *Theoretical Computer Science*, 402(2–3):190 – 198, 2008.

[81] N. Hanusse, E. Kranakis, and D. Krizanc. Searching with mobile agents in networks with liars. *Discrete Applied Mathematics*, 137(1):69–85, 2004.

[82] A. Hassidim and Y. Singer. Submodular optimization under noise. *COLT*, pages 1069–1122, 2017.

[83] T. Herman. Phase clocks for transient fault repair. *IEEE Trans. Parallel Distrib. Syst.*, 11(10):1048–1057, 2000.

[84] T. Higashino, Y. Katayama, T. Masuzawa, M. Potop-Butucaru, and M. Yamashita, editors. *SSS*, volume 8255 of *Lecture Notes in Computer Science*. Springer, 2013.

[85] B. Hölldobler. Recruitment behavior in camponotus socius (hym. formicidae). *J. of Comparative Physiology A*, 75(2):123–142, 6 1971.

[86] B. Hölldobler and E. O. Wilson. *The ants*. Harvard University Press, 1990.

[87] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. *FOCS*, pages 565–574, 2000.

[88] R. M. Karp and R. Kleinberg. Noisy binary search and its applications. *SODA*, pages 881–890, 2007.

[89] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *FOCS*, pages 482–491, 2003.

[90] A. Korman, E. Greenwald, and O. Feinerman. Confidence sharing: An economic strategy for efficient information flows in animal groups. *PLoS Comp. Biology*, 10(10), 2014.

[91] A. Kravchik and S. Kutten. Time optimal synchronous self stabilizing spanning tree. *DISC*, 8205:91–105, 2013.

[92] E. S. Laber and L. T. Nogueira. Fast searching in trees. *Eletronic Notes on Discrete Mathematics*, 2001.

[93] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[94] L. Le Goff and P. Soulier. Of ants and urns: estimation of the parameters of a reinforced random walk and application to ants behavior. 2014.

[95] C. Lenzen, T. Locher, P. Sommer, and R. Wattenhofer. Clock synchronization: Open problems in theory and practice. *SOFSEM*, pages 61–70, 2010.

[96] C. Lenzen, T. Locher, and R. Wattenhofer. Tight bounds for clock synchronization. *J. ACM*, 57(2), 2010.

[97] C. Lenzen and J. Rybicki. Efficient counting with optimal resilience. *DISC*, pages 16–30, 2015.

[98] C. Lenzen, J. Rybicki, and J. Suomela. Towards optimal synchronous counting. *PODC*, pages 441–450, 2015.

[99] S. Marras, R. Batty, and P. Domenici. Information transfer and antipredator maneuvers in schooling herring. *Adaptive Behavior*, 20(1):44–56, 2012.

[100] C. McDiarmid. *Concentration*, pages 195–248. Springer, 1998.

[101] S. Mozes, K. Onak, and O. Weimann. Finding an optimal tree searching strategy in linear time. *SODA*, pages 1096–1105, 2008.

[102] C. Musco, H. Su, and N. A. Lynch. Ant-inspired density estimation via random walks: Extended abstract. *PODC*, pages 469–478, 2016.

[103] K. Onak and P. Parys. Generalization of binary search: Searching in trees and forest-like partial orders. *FOCS*, pages 379–388, 2006.

[104] A. Pelc. Searching games with errors - fifty years of coping with liars. *Theor. Comput. Sci.*, 270(1-2):71–109, 2002.

[105] B. Pittel. On spreading a rumor. *SIAM J. Appl. Math.*, 47(1):213–223, 1987.

[106] N. Razin, J. Eckmann, and O. Feinerman. Desert ants achieve reliable recruitment across noisy interactions. *Journal of the Royal Society Interface; 10(20170079).*, 2013.

[107] G. Rieucau and L. A. Giraldeau. Persuasive companions can be wrong: the use of misleading social information in nutmeg mannikins. *Behavioral Ecology*, pages 1217–1222, 2009.

[108] P. Rigollet. High dimensional statistics. *Lecture notes for course 18S997.*, 2015.

[109] G. Roberts. Why individual vigilance increases as group size increases. *Animal Behaviour 51*, pages 1077–1086, 1996.

[110] S. B. Rosenthal, C. R. Twomey, A. T. Hartnett, H. S. Wu, and I. D. Couzin. Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion. *PNAS*, 112(15):4690–4695, 2015.

[111] S. Shah, R. Kothari, J. Dr, and S. Chandra. Trail formation in ants. a generalized polya urn process. *Swarm Intelligence*, 4:145–171, 06 2010.

[112] A.-S. Snitzman. Topics in random walks in random environment. *ICTP Lecture Notes Series*, 2004.

[113] D. J. T. Sumpter, J. Krause, R. James, I. D. Couzin, and A. J. W. Ward. Consensus decision making by fish. *Current biology: CB*, 18(22):1773–1777, November 2008.

[114] J. J. Templeton and G. L. A. Patch assessment in foraging flocks of european starlings: evidence for the use of public information. *Behavioral Ecology*, 6(1):65–72, 1995.

[115] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, pages 43–98, 1956.

[116] A. Xu and M. Raginsky. Information-theoretic lower bounds for distributed function computation. *IEEE Trans. Information Theory*, 63(4):2314–2337, 2017.